Where »less« is »more« – notions of minimalism and the design of interactive systems:

A constructive analysis of products & processes of human-computer-interaction design from a minimalist standpoint

Dissertation zur Erlangung des Doktorgrades
an der MIN-Fakultät
Department Informatik der Universität Hamburg

vorgelegt von Hartmut Obendorf

Hamburg 2007

Universität Hamburg

Genehmigt von der MIN-Fakultät Department Informatik der Universität Hamburg

auf Antrag von

Prof. Dr. Horst Oberquelle	Erstgutachter(in)/Doktorvater
Prof. Dr. Horst Oberquelle	Zweitgutachter(in)

Hamburg, den _____

Datum der Disputation 4.4.2007

Prof. Dr. _____
Leiter Department Informatik
(Prof. Dr. N. Ritter)

OVERVIEW

Table of Contents

Preface

I have only made this letter longer because I have not the time to make it shorter. —Blaise Pascal

This thesis assumes scientific transdisciplinary work is possible. Consequently, it is torn between a ‚rigid' logical tradition, which even in the purely mathematic proof is actually based on convention, empirical ‚evidence', that risks choosing chance over truth, and scholarly argument, where generalization is based on the transparency of personal bias. It presents a mixture of scientific work from the liberal arts, from design and from computer science. For some, it will not be detailed enough in its treatment of design, for others, principles of engineering will fall short. Yet, the marriage of different disciplines might appeal to those who believe it is the friction between different fields that makes the craft of designing interactive systems so interesting.

This thesis tries to differentiate meanings of simplicity—when we build something, we often want it simple, yet seldom know how. It is a thesis about minimalism, a concept from art, music and literature, and about its use in human-computer interaction. It documents a process rather than presenting a result, as the connection from the liberal arts to computer science is not immediate, and transferring a concept from one realm to another will need some argumentative support to hold the weight of argument.

We can well build (fairly) usable interactive systems. Experience tells us how to get things right—most of the time. Guidelines and rules help us to build good interfaces. Design processes, methods and techniques aid us in developing use-centred design. This thesis does not attempt to teach how to design better products in a better way; the author knows others with more practical experience, and is afraid to teach others on this subject. Instead, this thesis establishes perspectives on the why. Why do some products work better than others? Why do guidelines lead to simple designs? Why do methods work?

Minimalism is part of this answer. Reduction lies at the heart of design. Identifying key users and key tasks and limiting immediate functionality to the essential produces simple, yet powerful products. Selecting, structuring and modularizing functionality, fitting tools to a specific task, configuring complex work environments, and crafting tools for useful misuse are all important parts of design. Reduction implicitly guides designers, design rules and design methodology. Minimalism is used here to differentiate between these design goals and design activities, it is used as a tool to identify the benefits of reduction, and the trade-offs involved. But it also makes explicit the dangers and shortcomings of an approach following ‚less is more', of less becoming a bore.

I thank Horst Oberquelle, because he made this thesis possible, Kaj Grønbæk, because he made me improve upon it. And I thank all others who made developing this thesis the joyful and fulfilling experience that it was.

Thank you.

1 Designing for an Age of Complexity

In today's increasingly complex society, the „digital revolution", creating new tools and new ways of working, has contributed additional layers of complexity to our lives—although it set out to make work faster and simpler by building machines that help us calculate, and although mastering complexity through abstraction and systematization has been one of the constitutive goals of computing science.

New questions were raised by the dissemination of computer artifacts throughout society and, with the advent of informatics as a research discipline that complements computer and computing science, focus shifted from the engineering view of reducing complexity for the designing engineer to reducing complexity for the end user and ensuring his being in command of the procedures employed. Inseparably bound together, both complexities cannot be analyzed in separation; many problems that surface when systems are put into use are caused by engineering complexity adding to application complexity.

Reducing this complexity is the purpose of the use-centered design of interactive systems. Ideals, such as ‚simplicity' play an important role in the self-conception of this discipline. Design aims to create interactive systems so simple they are no longer recognizable as systems, but fade into the background, quietly enhancing our abilities. What, however, *is* simplicity? The meaning of the ‚simple' changes with both protagonist and subject, with both source and focus of perspective. Common to all notions of the simple is only their *relative* nature—they all refer to some type of *reduction*.

In this thesis, the *notion of minimalism* is proposed as a theoretical tool supporting a more differentiated understanding of reduction, and thus forms a standpoint that allows to define aspects of simplicity. Possible uses of the notion of minimalism in the field of human-computer interaction design are examined with both theoretical and empirical focus, yielding a range of results: Minimalism defines a refreshingly novel, and possibly even useful standpoint for design analysis. As the empirical examples demonstrate, it has also proven to be a useful tool for generating and modifying concrete design techniques.

1.1 Who should read this Thesis?

As a thesis comprising both theoretical and empirical parts, different audiences should read this thesis differently. Practitioners, such as *software engineers*, *designers*, or *usability experts* are similarly targeted as those who are more interested in the theoretical framework, be it from the perspective of art, or from human-computer interaction.

The *software engineer* will want to parse the definitions of minimalism (3.3), and then directly skip to the software development methods defined in this thesis (6). She might then want to read more about the defined notions of minimalism in the closing discussions of related perspectives (4.7) and designs (5.5) before jumping to the conclusions (7).

The *designer* will find most value in the discussion of real-world designs (5), and might want to refer to the more rigid definition of minimalism (3.3) later. The Minimal Crit Game (6.1), and—should he often work with engineers—the other techniques developed (6.2.3 and 6.3.2) might provide inspiration for his design practice.

The *usability expert* will find the discussion of existing norms, guidelines, and expert lore in terms of minimalism (4) to provide a fresh perspective on her own practical experience. Examples from both analog and digital designs (5) serve to deepen her understanding of the different notions of minimalism used in this thesis (3.3).

Readers seeking a deeper understanding of the *minimalist standpoint* and its development will find manifold sources for the definition of minimalism in human-computer interaction in the discussion of the sources underlying this work in art and music history (2). They will also want to follow the derivation of the notions of minimalism for human-computer interaction and understand the limits of the scope of this work (3).

For all readers, this first chapter sets out to roughly sketch the ideas leading to (1.2) and defining (1.3) this thesis before providing an overview of its individual chapters (1.4).

1.2 Motivations for Minimalism in HCI

The skeptical reader may ask: Who needs the notion of Minimalism? And is it necessary to examine and explain *simplicity*? In practice, simple systems can be created without the need for a theoretical conception: experience, or even trial and error will eventually produce simpler, better solutions. Fieldwork, user modeling, and user testing can help to identify these, and successful engineering techniques exist to realize them.

It is not only the scientific interest of understanding simple design that motivated this thesis. Clearly, random hillwalking will get results, but it is not only much more gratifying, but also far less expensive if simplicity can be planned for. The practitioner should be able to consciously choose her tools, and make informed design decisions reflecting those aspects of simplicity her design tries to achieve. To this end, this thesis defines an ideal for design (1.2.1), focuses on reduction as a technique (1.2.2), and draws on the notion of minimalism to differentiate understandings of simplicity (1.2.3). The resulting minimalist terminology helps to understand qualities of designs, and how these qualities are created in design techniques and processes.

1.2.1 Machine Beauty = Power + Simplicity

There is a continuous and controversial argument about the focus of design—usefulness or beauty, and about the influence of aesthetics on usability or usefulness of artifacts. While the field of human-computer interaction has long been focused on ‚fitness for tasks', or functionality, the discussion currently seems to bend towards embracing ‚funology', or the anticipatory specification of use experiences (e.g. Norman, 2002a).

David Gelernter introduced a concept that dissolves this constructed dualism: for him, ‚machine beauty' encompasses both the usefulness of an artifact, and the joy its use will generate. Machine beauty is created through the combination of power and simplicity. Beauty thus is more than skin deep, more than a feeling to be measured; it is an explanation for the intrinsic quality of ‚tool'-ness that a „good" design emits (Gelernter, 1999). The term ‚machine beauty' also highlights that the strength of great tools comes through a combination of abilities and their ease of access. While human-computer interaction methodology customarily used efficacy (or effectiveness) and efficiency as measures for usability, Gelernter stresses that machine beauty is created through more than power or simplicity in isolation.

A concentration on power and simplicity is nothing new in design: The Bauhaus school of design followed the motto of ‚form follows function' and aimed to abolish ‚superfluous' decoration. This led to an explicit focus on functionality. However, it turned out that this simply rephrased the design question as ‚what is function?', and that the differentiation between decoration and function is seldom easy (cf. 5.5.1). A central objective for this thesis is thus the examination of the question ‚what is simple?' And due to the relative nature of simplicity, the question of degree—'how simple must it get?' follows immediately on its heels.

1.2.2 Reduction – Give Up or Gain?

Less is more. This is a mantra that has been followed by many a designer. Yet, it does not answer the two questions given above, it does not tell *what exactly* should be less, and *how much less* is still more. And is not sometimes also more more?

In this work, a method—or rather a procedure—for producing ‚machine beauty' is examined in more detail: reduction. It is important to keep in mind during the lecture of the following pages that the central proposition ‚less is more' should not be understood singularly as ‚simple is better'. Although there may be some merit in this weaker statement, ‚less is more' also acknowledges the need for power, the quest for functionality that can be observed in almost any digital artifact. It should also be kept in mind that other methods and tools exist for the designer or the engineer, reduction is not always useful, and sometimes dangerous for the product. Yet again, while universal machines are theoretically elegant solutions, tools for the real world need not be Turing complete. While the Swiss army knife has many functions, it is inferior to specialized tools whose power often accrues out of limitation; in other words: not everything constructed by a computer scientist should be a computer, lest universal power will be hidden by universal complexity.

Reduction as such often seems to be deceivingly easy as the result of a reductive process appears to be simple and ‚less' than before; all one needs to do is strip off the superficial ornamentation, ask for less unnecessary interaction, concentrate on key tasks and thus provide less superfluous functionality, or introduce a service layer that conveniently defines away the underlying complexity of the service infrastructure.

However, reduction is almost always hard work: ornamentation is difficult to discern from ‚nice' design, established procedures must be followed for the sake of consistency, and breakdowns all too often expose the underlying infrastructure, making repair impossible as no interference with the hidden complexity is possible. And even the introduction of reduction as an ideal into the design process is extremely difficult: more pre-planning might be necessary, the benefit of spending extra money to perhaps lower hidden costs that would surface later is difficult to prove. Development processes are often driven by features, internally motivated by the developers—easily observable in open source projects, and externally by marketing demands—reduction is hard to sell as a feature.

1.2.3 Minimalism: Borrowing the Extreme from the Arts

Returning to the two questions a foray into reduction is confronted with, that of degree, and that of direction, this work tries to arrive at a systematic viewpoint to explain why products and processes strive to reach ‚good design' through reduction: what exactly is the right target for reduction—what methods exist to find reducible aspects of a design, where is the place to start the search?

To find some orientation support, the extreme was chosen as a model in this work: the phrase ‚less is more' is not only used in design, it is closely linked to the term minimalism, which initially referred to an art movement in the 1960s and was later successfully applied to music, literature, dance, architecture, design and a multitude of other disciplines. Minimalism always denotes an extreme case of reduction, and a central proposition of this work is that we can learn from looking at these extremes.

Although simplicity as a value is widely accepted, simple systems are rare—most instead threaten to burst with complexity. Failing to understand why this was so, this thesis set out to find help for understanding this paradoxical state—in the liberal arts. Art is an arena where values that are becoming important are often displayed and discussed with extreme violence and clarity, long before they become commonplace in other discipline. In the words of the director Peter Greenaway „today, many perceive painting as something that is both remote and insignificant. That is a tragical mistake. Painting is always ahead where sociological or philosophical developments are concerned. Take a look at the 20th century: all philosophical movements began with painting. Cubism, Surrealism, Minimalism, Structuralism, and so forth. A new way of thinking about the world manifests itself always in painting. For me, painting is leading all other arts." (Greenaway, & Rauterberg, 2005, retranslation mine)

Art provides a manifestation of new ways of thinking about the world—definitions of perspectives from which things look different. Through its avant-garde nature, analyzing

art can either help scouting trends, or make use of the longer experience that critics and historians have acquired examining a development. The latter approach is followed in this thesis: minimalism has been practiced and analyzed since the 1960s, enough time for a thorough analysis of the minimalist standpoint. Since then, many disciplines have adopted minimal terminology as a description for existing trends. This thesis aims to do the same for computer science, or more specifically, interaction design.

The approach taken tries to bring up answers that help to identify and set the direction of reduction, and illustrate trade-offs in degree. Minimalism is a better guide than simplicity as it is not necessarily desirable. It can produce boring, inadequate, even dangerous results. However, making use of the extreme as a model for design makes ‚extreme' traits visible in everyday products, and allows designers to find the right degree of reduction.

1.3 Minimalism in a Nutshell

It took the better part of three years to develop and understand the four notions of minimalism that are described in this thesis. While a complete reading will take less time, this section presents a first overview of subject and method of this thesis to further shorten the waiting time for the reader. Details on the derivation of the four notions of minimalism are omitted, but an example demonstrates that they differentiate forms of simplicity.

1.3.1 Four Notions of Minimalism, their Relationship and Design

While the initial intention behind this work was to define a single, unified standpoint for ‚minimalism', the studies of the literatures in art and music history quickly made it clear that there existed no consensus about the signification of the term. Minimalism is a term used to describe art and music that is at the same time very similar and very different. Although this was disappointing at first sight, it became quickly clear that this multiplicity of perspectives is a virtue rather than a defect: it allows the disambiguation of different perspectives on minimalism, and thus of different kinds of simplicity.

Four different notions of minimalism that were observed to recur in the different literatures in art and music are introduced to the design of interactive systems. This transdisciplinary transfer is based upon an extensive analysis of the critical discourse in both art and music that began in the 1960s, and is still active today. Although the protagonists of minimalism are diverse in both their conception of reduction and their judgement of artistic qualities, five concepts repeatedly surface in the different literatures: a *minimality of means*, a *minimality of meaning*, and a *minimality of structure*, the *use of patterns*, and the *involvement of the recipient* in the work of art.

For the design of interactive systems, four notions of minimalism were identified drawing on these common qualities of the minimal. The four notions of minimalism focus on the *function*, *structure*, *architecture* and *composition* of the *interface*. The choice of words is deliberate: the former two directly describe aspects of the concrete design, while the latter two point towards more transient aspects of the design that are determined by the construction method and the introduction into the work context. Functional and composi-

tional minimalism focus on the *use* aspect of the tool-in-context, while structural and architectural minimalism highlight how functionality is *accessed* by the user.

Table 1: The framework for analysis consists of four notions of minimalism.

tool	1 Functional Minimalism	2 Structural Minimalism
context	4 Compositional Minimalism	3 Architectural Minimalism
	use	access

To illustrate the relationship of the minimalist terminology with existing terms in computer science, Figure 1 reproduces a diagram often used to describe the architecture of software: a presentation layer handles interaction with the user, a logic layer does invisible processing, and a data layer allows the storage and manipulation of data. Variations typically split one of these layers in two, e.g. the presentation layer to model lightweight distributed clients, or the logic layer to allow a distinction between presentation logic and data logic. In this thesis, neither ‚architecture' nor ‚structure' relate to the inner construction of a software system. Instead the focus is on the interface of a design: assuming a designer's perspective, the design of the presentation layer is central for the use experience. Subjacent system layers become relevant only as they determine the design of the interface, and the concrete details of interfacing the different layers are no central issue. Thus, the discussion explicitly avoids identifying different layers of a system, highlighting that even though the distinction between different layers is useful to engineer a product, some design decisions for the interface reach deep down into the conception of data and logic.

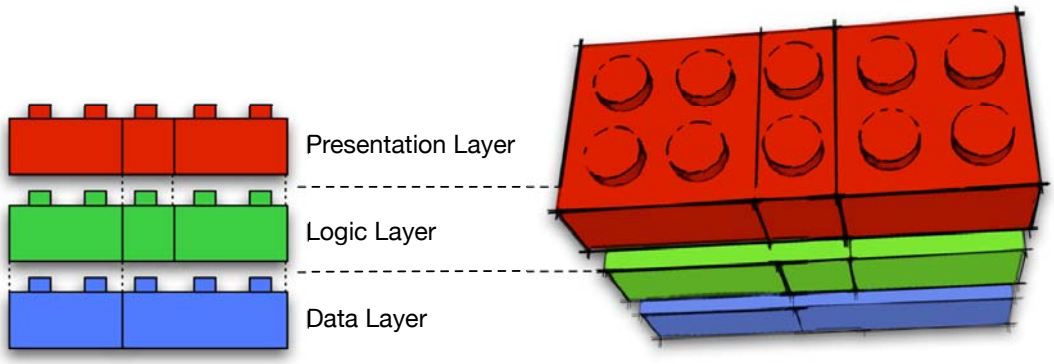


Figure 1: The formal and the designer's perspective: discussion in this thesis focuses on the interface.

1.3.2 An Example Use of the Notions of Minimalism for Product Analysis

A concrete example might illustrate these rather abstract definitions, although it will fall short of explaining them in depth—as the different notions of minimalism aim to highlight different aspects a design that can be made minimal, a single example will only partly excel in each area of minimalism. Nonetheless the iPod Shuffle (Figure 2), a music player that was introduced by the market leader Apple, offers a stunning example for how complex technological devices can feel simple (compare 5.3.4).

Figure 2: The iPod Shuffle, introduced in 2005, has only few buttons, and lacks a display.

Even at first sight, the iPod Shuffle is very reduced in terms of *functionality*. That does not mean that it is not technically sophisticated: it is able to play a wide variety of different media types, and provides an audio signal of good quality. Its functional minimality lies within the interface—except for the playing, it can only skip songs and change the volume. Although its internal hardware would support a display, recording audio, and a radio function (Williams, 2005), the iPod Shuffle does without them.

The interface *structure* of the iPod Shuffle is consequently very simple. There is no display, and no menu system. Each function is mapped directly to an interface element. This allows the user to immediately grasp control of the iPod Shuffle, and even using it without looking becomes possible. However, there is one twist to the Shuffle that makes its behavior more complex: it has two modes, one where a pre-defined playlist is played from top to bottom, and then repeated ad infinitum, and one where the playlist is shuffled before each pass. Yet, except for changing the sequence of the playlist, the mapping of functionality is identical in both modes.

Figure 3: The iTunes software is used to compose and select playlists for the iPod Shuffle.

The functional simplicity of the iPod Shuffle is, however, not the only reason that its access structure is so simple. An important aspect of the iPod Shuffle is that is was never designed to be used alone. Instead, the iTunes music player and library manager was extended as to provide functionality that competitors boast with (Figure 3). Apple had decided that a desktop computer with its large screen and rich input channels is far superior for tasks such as selecting music tracks and composing playlists. And thus, the responsibilities are shared between iTunes and the iPod Shuffle: iTunes is used to select and arrange playlists, the iPod Shuffle simply plays. This sharing of responsibility across different tools that combine to form the overall interface is referred to by *architectural minimalism*: the combination of several simple tools simplifies the handling of complex tasks.

Finally, the iPod Shuffle is also minimal in its *composition*. Again, this refers to its interface and does not mean that it is not engineered thoughtfully. Instead, compositional minimalism marks a maximization of possible uses that the iPod Shuffle can be put to. By designing a functionally minimal device, Apple does not restrict the use of the player, and except for the interaction of iTunes and the player itself, there is no strict workflow to follow. Its functionally, structurally and architecturally minimal design helps the user to feel in control. Thus, the iPod Shuffle offers itself to the user for appropriation. There are examples for constructive uses, such as the use of the player to rent e-books to library users (Stephens, 2005), or as an instrument for students to rehearse music pertinent to their studies—and for faculty as a first step towards new technology use (Calhoun, 2005). Other examples describe decorative uses—here, the users literally make the technology their own, and recreate the design in a way that is appropriate for their specific use (Figure 4).

Figure 4: Examples for decorative misuse of the Apple iPod Shuffle found on the Web.

The iPod Shuffle was simpler than its competitors, and it offered less features. On the market less than six months, it captured a 58% share among flash music players (Gibson, 2005). The short discussion here has tried to illustrate that the simplicity of the iPod Shuffle originates in a number of different aspects in its design, and although many more

examples are necessary to sharpen the minimalist terminology used here, this brief example has demonstrated that the notions of minimalism are applicable to digital design.

1.3.3 Minimalism, Products and Processes

An underlying assumption for this thesis is that a wide range of products today already implicitly follow minimal values, or is designed with reduction as a motive. While I know of no scientific study that might prove this argument, observation of current mature electronic products not targeted at the early adopter demonstrates a rise of ‚simplicity' and an increased stress on ‚usability' vs. ‚abilities'. Also, some support is lent by slogans like ‚sense and simplicity' (Philips 2005) or new brands like ‚SIMply' (Vodafone 2005) that advertise simplicity as a key ‚feature'.

Minimalism is used in this thesis to differentiate between possible meanings of simplicity. This is done first for design products, where minimal qualities are identified that match with one of the proposed four notions of minimalism. The products covered in the fifth chapter of this thesis include mass-market hardware products like the Nokia 3310 cell phone series, the Apple iPod family, but also current consumer software products like the iLife series that try to strip down full-featured products to fit the consumer market. From the personal experience of the author, the HyperScout web enhancement system, and the CommSy CSCW system are examined. The application of the developed categories of minimalism to example products demonstrates the usefulness of minimalism as a standpoint for analysis, and it also helps to sharpen the theoretical framework as the abstract notions are connected with example qualities.

The thesis continues with a more constructive part, where the notions of minimalism are put to action in the creation of new designs; this was partly motivated by the personal involvement in the development and evaluation of the latter two examples. Consequently, the question of minimalism in development processes is dealt with in more detail. The usefulness of minimalism is demonstrated on three, increasingly abstract levels: First, the Minimal Design Game as a design technique is developed using the identified design qualities. Second, a design process is composed based on a minimal analysis of its components, such as Scenario-Based Development (Rosson, & Carroll, 2001) and agile development processes (Beck et al., 2001). Specific scenario techniques are selected to accent essential information in the implementation agenda. Third, Value-Based Design as a genre of design techniques is identified in the CommSy development process, describing its gradual focus shift from functional and structural minimalism to architectural and compositional minimalism.

1.4 The Structure of this Thesis

This chapter introduced the application of notions of minimalism to the design of human-computer interaction as a guiding theme for this thesis. Following a short discussion of methodological problems and choices made in this work, you are now reading an overview of what awaits you in the individual chapters.

This thesis is divided in two parts: The first, tracing the history of minimalism and narrowing down a theoretical definition to four kinds of minimalism and their relationship to existing work in chapters two, three and four, prepares the ground for the second part, where a practical discussion of minimalist traits in information artifacts and minimalist goals of development techniques tests and validates the use of minimalism as a perspective on interactive computing in chapters five and six.

In chapter two, examples of works in art, music and literature that were collectively described with the label of minimalism by contemporary criticism and art history are revisited. This chapter follows a historical rather than a conceptual order and aims not at a single definition of minimalism, but instead tries to illustrate both the breadth of concepts underlying works characterized as minimal, and the recurrence of attributes of minimal art in different disciplines.

In chapter three, based on these shared aspects of minimalism, four principles, namely functional, structural, constructional and compositional minimalism, are introduced. Before these concepts are defined in the context of the use-centered design for interactive systems, the scope of this transfer is set as different possible interpretations of minimalism that are not covered in this thesis are illustrated.

In chapter four, they are set in relation to existing concepts in the field of human-computer interaction. Existing related work, such as the practice of industrial design (Dreyfuss, 1955), international usability standards, or Norman's concept of appliances (Norman, 1999b) are used to demonstrate the minimalist viewpoint.

In chapter five, introducing the second part of this work, the minimalistic principles are put to the test as different products of design are examined for their minimalistic qualities and possible problems created by reduction. While all examples—a mixture of research prototypes and commercial applications—apply reduction in some way and at some point, there is a surprising variation in results.

In chapter six, the analysis is carried forward to design processes. Building upon the minimal qualities defined in the previous chapter, a design crit technique is developed. Minimalism is also used to analyze existing development techniques, namely the use of scenarios, and to identify which techniques need be included in a development process for a given minimal aim. To demonstrate the abstract use of minimalism, an existing development process is analyzed, identifying a genre of development techniques.

Finally, in chapter seven, the product and process threads are reevaluated and the dimensions of minimalism are evaluated for their usefulness in analyzing human-computer interaction design. A critical reflection of the ‚mental apparatus' that is established by the four notions of minimalism also shows limitations of the approach.

2 In Search of ‚Minimalism' – Roving in art history, music and elsewhere

The following part is designed to give an overview about the meanings that have been discussed under the label ‚Minimalism'—to prepare the ground for a discussion of meanings that ‚minimalism' might assume for the design of interactive systems.

'Minimalism' is a term of many uses and diffuse boundaries. Time and protagonists of the exact coinage of the phrase differ according to perspective: The musician Steve Reich credited the critic Michael Nyman with the invention of the term and its application to music in the seventies (Nyman, 1974); his colleague Philip Glass took Tom Johnson for the originator of the term. However, art critic Barbara Rose had already applied the term (along with her own conception, „ABC art") to the music of Young and the choreography of Judson's Dance Theater in fall 1965 (Rose, 1965a). She then credits Wollheim's January 1965 article in Arts Magazine (Wollheim, 1965) with the formal introduction of the term. Yet, this again is subject to debate, as—taking usual lead-time into account—two articles by Rose herself and sculptor Donald Judd published in February 1965 (Rose, 1965b; Judd, 1975, 35) might have been written before. Looking further back, Judd had used the term writing about Morris' work in March 1964 (ibid., 118), and, even in March 1960, described Paul Feeley's „confidence in the power of the minimal" (ibid., 17). Furthermore, David Burliuk described the painting of John Graham as „Minimalist" in 1929, noting „Minimalism … is an important discovery that opens to painting unlimited possibilities" (cf. Strickland, 1991, 19).

One can safely assume only that the term ‚Minimalism' earned public attention during the 1960s and gradually replaced other, competing labels, such as the aforementioned „ABC art", and „reductive art" (Rose, 1965a), „literalist art" (Fried), „structurist" sculpture (Lippard, 1967), „object sculpture" (Rose, 1965a), „systemic painting" (Alloway, 1966), „specific objects" (Judd, 1975), „unitary forms" (title of Morris' 1970 San Francisco Museum of Art exhibition) or „unitary objects" (Sandler 1967)—to mention but a fraction of the terms used only within art criticism.

'Minimalism' denotes the minimal, often in relationship to contrasting practices, and according to context, the focus of the minimal changes. Until today, no single definition has been agreed upon, and some authors even deny its possibility. To clarify the relative nature of the term, some recent examples are restated here:

> Minimalism – „an artistic tendency whose »organizing principles« were »the right angle, the square and the cube … rendered with a minimum of incident or compositional maneuvering«" (Colpitt, 1990, 1).
>
> Minimalism – „a movement, primarily in postwar America, towards an art – visual, musical, literary, or otherwise – that makes its statement with limited, if not the fewest possible, resources." (Strickland, 2000, 7)
>
> Minimalism was „the last of the modernist styles" and thus „a transition between the modern and the postmodern." (Levin, 1979)
>
> "The only reason that I did any writing … is really the fact that the critics had not understood things very well. They were writing about Minimal Art, but no one defined it … People refer to me as a Minimal artist but no one has ever defined what it mean or put any limits to where it begins or ends, what it is and isn't." (Sol LeWitt in Cummings, 1974)
>
> „This book … views minimalism neither as a clearly defined style nor as a coherent movement that transpired across media during the postwar period. Rather, it presents minimalism as a debate … that initially developed in response to the three-dimensional abstraction of, among others, Donald Judd, Carl Andre, Robert Morris, Dan Flavin, Anne Truitt, and [Sol] LeWitt during the period 1963-1968. … Minimalism was a shifting signifier whose meanings altered depending on the moment or context of its use … a field of contiguity and conflict, of proximity and difference … a new kind of geometric abstraction". (Meyer, 2001, 1)

Definitions of minimalism range from the specific to the broad, from the temporal to the technical, from the impossible to the dialectic—depending on the motive of the author. Some of these definitions are more difficult to use as a basis for this work, as specificity complicates translation into a different field, temporality limits the meaning of minimalism to a single movement. The more technical definitions promise the viability of comparison with other „movements", and the dialectic analysis proposed by Meyer shows potential for understanding the dynamics of artists' self-definitions and their interactions with other practitioners, thus elaborating the essence of the term.

Yet, as the purpose of this overview is not to define minimalism within a particular context, but rather illustrate the broad range of meanings that was denoted by ‚minimalism' from the conception of the term sometime in the 1960s until the present day, a less strict approach is followed: For clarity's sake, the sequence of appearance is oriented on genre and temporal order; whenever deemed appropriate, a definition that was useful within a narrow context is mentioned and set in relation with contrasting positions. Although there are some exceptions, such as the concurrence of Young's and Glass's first musical ex-

periments and the development of Minimal sculpture in the New York art scene, this approach works quite well, as minimalism was first used to describe painting, then sculpture and film, then music and eventually literature before the term was applied to subjects only marginally connected to art. The validity of applying the term ‚minimalism' to a range of styles that span different disciplines is certainly motivated by this work's aim of discerning meanings that might be transferred analogously to human-computer interaction, yet the connection of the different minimalisms, particularly of those in dance, music and art is strengthened by the history of personal relationships of the artists that created their works with something that might be called a common spirit.

2.1 Minimalism in the Arts

Minimalism in the fine arts originated in painting, and was later continued in sculpture. Its different protagonists created very different artworks, and followed different conceptions of reduction. In reduction, they focused on topics such as color, material, and structure. An important consequence was the establishment of the object-character of artworks, and the use of ready-made materials. While this was first developed in painting (e.g. in the works of Reinhardt), it later culminated in sculpture (and was made explicit by Judd). Consequently, the overall Gestalt of an artwork evolved into a central aspect of minimalist art, and relationships that extended beyond the object, and included the spectator, became relevant to minimalist artists.

2.1.1 Rauschenberg, Klein and Newman: The Birth of Minimal Painting

Perhaps the first ‚Minimal' paintings were created by Robert Rauschenberg in 1951, six works composed of from on to seven panels of rolled white enamel paint, pioneering the use of housepaint on unprimed canvas and preceding the black paintings of Frank Stella by seven years. The simple whites were produced with a lack of painterliness that leaves little room for a more extreme reduction; in contrast to Robert Ryman's monochromes of the 1960s, Rauschenberg did not even concern himself with the calligraphy of the brushstroke (Figure 5). He describes his paintings with great avidity, however: „They are large white (one white as God) canvases organized and selected with the experience of time and presented with the innocence of a virgin. Dealing with the suspense, excitement, and body of an organic silence, the restriction and freedom of absence, the plastic fulness of nothing, the point a circle begins and ends. They are a natural response to the current pressures and the faithless and a promoter of institutional optimism. It is completely irrelevant that I am making them. Today is their creator" (quoted in Ashton, 1982, 71).

As Strickland (2000, 28) notes , „the paintings were not, however, as God- or nothing-like in practice as in theory" – they were instead viewer-interactive from the beginning: „I always thought of the white painting as being, not passive, but very—well, hypersensitive … so that one could look at them and almost see how many people were in the room by the shadows cast, or what time of day it was" (Tomkins, 1981, 71). Besides the extreme reduction of means, we also find the inclusion of the viewer as one of the prime characteristics of these works; their reception differs with a different setting or a different frame

of perception as they hold only a minimal amount of pre-defined information themselves. In contrast to the monochromatic works of Josef Albers (a retrospective can be found in Albers, & Weber, 1988), Rauschenberg broke with the focus on the relationships within the painting, he shunned the use of hierarchical composition and figure/ ground contrasts that Albers had concentrated upon (Albers, 1963): „I had been totally intimidated because Albers thought that one color was supposed to make the next color look better, but my feeling was that each color was itself" (Rauschenberg in Rose, 1987, 37-38).

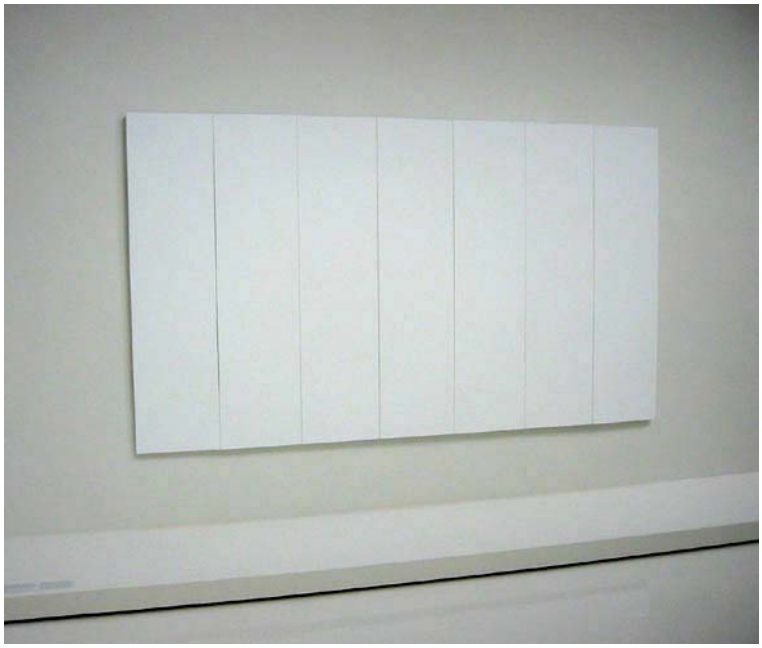


Figure 5: Rauschenberg: White Painting (1951).

Yves Klein, a world-class provocateur and an art celebrity for the last five year of his life (1928-1962), also laid claim on the creation of the first monochromes, although his proofs are rather insufficient (Strickland, 2000, 33-34). He contributed the ultimately minimal exhibition to art history, at the Iris Clert gallery in Paris in 1958 – described in the invitation as „the lucid and positive advent of a certain realm of sensitivity … the pictorial expression of an ecstatic and immediately communicable emotion" (quoted in ibid., 35). The gallery in question was empty; Klein himself had painted the immediately communicable walls with white paint. Klein is best known for his monochromes painted in ‚International Klein Blue' (or a deep royal blue). He chose blue over other colors: „Blue has no dimensions, it is beyond dimensions, whereas other colors are not. They are prepsychological expanses, red, for example, presupposing a site radiating heat. All colors arouse specific associative ideas, psychologically material or tangible, while blue suggests at most the sea and the sky, and they, after all, are in actual visual nature what is most abstract" (Osborne, 1988, 295). In addition, Donald Judd stressed in 1965 the influence of absence of compositional space: „Almost all paintings are spatial one way or another. Yves Klein's blue paintings are the only ones that are unspatial, and there is little that is nearly unspatial, namely Stella's work" (Judd, 1975, 182).

Figure 6: Barnett Newman: Adam (1951).

Sometimes counted among the minimalists, Barnett Newman is usually considered one of the most influential predecessors to minimalism, as his work determined the visual appearance of minimalist painting despite him maintaining contradicting theories (Strickland, 2000, 55): he tried to limit references to the work itself, noting „I hope my work is free of the environment" (Newman, O'Neill, & McNickle, 1990, 272), and rejected the void created by reduction in e.g. Klein's drawings (ibid., 249). Instead, his zip paintings (Figure 6), the label stemming from the regularity of dividing the canvas into areas of different color, are not targeted at reduction, but rather try to create a unity of the whole artwork: „I feel that my zip does not divide my paintings … it does the exact opposite: it unites the thing" (ibid., 306).

2.1.2 Reinhardt: art-as-art

Ad Reinhardt was perhaps the purest minimalist painter, Sol LeWitt called him „the most important artist of the time" and producer of „the most radical art" (Zelevansky, 1991, 19) and Rosenberg identified him as „the intellectual pivot" of the Minimalist movement (Rosenberg, 1964). He most vividly contrasted Klein and Rauschenberg in character: against their theatricality he set what was described as an „eremitic aesthetic and […] near-obscurant technique" (Strickland, 2000, 40) and what Reinhardt summed up in 1962 as „The one thing to say about art is that it is one thing. Art is art-as-art and everything else is everything else. Art-as-art is but art. Art is not what is not art" (Reinhardt, 1975, 53). Art-as-art describes the total rejection of context and creator as meaningful aspects of the work – a piece of art is only what it is, no more, no less. It is directed against the mystification of „simple" art and the displacement of visual abundance by elaborate explanations and interpretations. He also vehemently refuted attempts to relate his work

to the action painting of e.g. Pollock: „I suppose there is always an act or an action of some kind. But the attempt is to minimize it. There are no gymnastics or dancings over painting or spilling or flipping paint around" (ibid., 13).

Reinhardt tried to abolish all external references from his titles as from his paintings, most of which are simply signed ‚Untitled', ‚Abstract', or ‚Red/Blue/Black Painting' – often on the back of the canvas, not always including the date. His aim was to create „the last paintings which anyone can make" (Lippard, 1981, 158); he tried to achieve maximum reduction. His „Twelve Technical Rules" read as follows: „No texture … No brushwork or calligraphy … No forms … No design … No colors … No light … No space … No time … No size or scale … No movement … No object, no subject, no matter. No symbols, images, or signs. Neither pleasure nor paint. No mindless working or mindless non-working. No chess-playing" (Reinhardt, 1975, 205f).

The reduction in Reinhardt's works did not come suddenly, it was a slow and steady development towards the minimal: „At many stages along the way, it seemed as if he had gone as far as he could go, but each year he reduced the elements in his art a little more" (Bourdon 1967). The premier example for his Minimalist works is the ‚black' paintings—he was far from being the first to explore black-on-black, as e.g. Rodchenko had executed a pure black in 1919, yet his approach is unique. The ‚black' paintings are not black at all, on close scrutiny, they are composed of deep blue, red and green pigments in various mixtures; he drew diagrams before executing his paintings that show that most of them consist of a nine-squares-within-a-square structure, with three different tones marked as ‚B', ‚R' and ‚G' occupying the corners, the middle row and the middle upper and lower square, respectively (Zelevansky, 1991, 21).

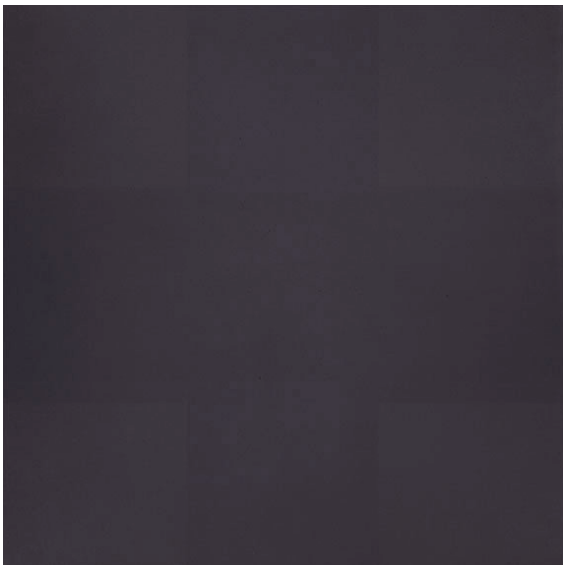


Figure 7: Ad Reinhardt, Abstract Painting (1960-66). Solomon R. Guggenheim Museum.

Reinhardt used hues of blue, red and green so dark that various critics have noted that the normal museum visitor does not, and literally cannot see these paintings: the ocular

adjustment necessary to perceive the shades of color takes time the casual viewer does not spend on the paintings; thus, they are often dismissed as a joke (Figure 7). For the darkest, last paintings, only flashlight photography reveals the cruciform shape; Strickland notes, „from a phenomenological perspective, one might argue that they [the photographic reproductions] invent them" (Strickland, 2000, 49). Although the suggestion has been made frequently, the reason for the choice of color, or rather non-color is arguably not a religious one, as is the shape only superficially reminiscent of a cross: „The reason for the involvement with darkness and blackness is … an aesthetic-intellectual one … because of its non-color. Color is always trapped in some kind of physical activity or assertiveness of its own; and color has to do with life. In that sense it may be vulgarity or folk art or something like that" (Reinhardt, 1975, 87). He thus extends Klein's dismissal of all colors except blue to a complete dismissal of color for its relation to life and thus its potential ‚folk art' character; again, art is only art-as-art.

Beginning with the choice of the square as the simplest geometrical form as the basis of the structure of the painting, and then further reducing the structure to the point of indiscernability by minimizing value contrasts, Reinhardt tried to achieve the greatest abstraction in painting. He even refrained from taping his canvases to avoid the „miniscule ridge which might define the border between squares and … destroy the flatness of the surface" (Strickland, 2000, 49). His paintings tried not only to be non-relational in their inner composition, but also to be non-referential; he went so far as to drain the oil from his paints to reduce reflections: „There should be no shine in the finish. Gloss reflects and relates to the changing surroundings" (Reinhardt, 1975, 207). He did not view the individual canvas as an almost sacred source of inspiration, as the Abstract Expressionists did; although he employed painstaking brushwork to remove all traces of brushwork from his paintings, he has more in common with the Minimalist sculptors who ‚only' planned their works before they were industrially executed: „When one of his works was damaged, and the museum asked him to repair it, he offered to substitute another version. »But you don't understand, Mr. Reinhardt,« he was told. »Our Committee especially chose this one.« »Listen,« the painter answered, »I've got a painting here that's more like the one you've got than the one you've got«" (Hess, 1963, 28).

2.1.3 Stella: To See What Is There

Between art critics, an argument developed in retrospect whether Reinhardt should really be considered minimal as he simply further reduced „pictorial asceticisms" from the past. Reinhardt was compared with Frank Stella who was more concerned with pure surface. Among others, Rosalind Krauss tried to differentiate them in her explicitly revisionist review: „it soon seemed obvious [in 1963] that what [Reinhardt's and Stella's ‚black' paintings] had in common was, nothing" (Krauss, 1991, 123).

Without doubt, however, is Frank Stella another of the central figures of Minimalist painting; he presented his first exhibition at the Museum of Modern Art an almost unbelievable two years after his college graduation, yet it took art critics until 1965 to agree that „the uncommon strength and integrity of Stella's young art already locate him among

the handful of major artists working today" (Rosenblum, 1965). In contrast to the obscurity of Ad Reinhardt's near-black paintings, Frank Stella's monochrome works that he arrived at in 1958 after experimenting with alternating bands of color, specifically red and black, expose both the crude technique and the unmodified enamel housepaint used: „I knew a wise guy who used to make fun of my painting, but he didn't like the Abstract Expressionists either. He said they would be good painters if only they could keep the paint as good as it is in the can. And that's what I tried to do. I tried to keep the paint as good as it was in the can" (Glaser, 1966, 157). In contrast to the perfectly symmetrical and carefully executed paintings of Reinhardt, Stella's freehand technique resulted in an irregularity of the outlines of the bands and in an inexact placement (Figure 8). The intent of Stella's works was to create truly non-relational paintings: „The black bands and interstitial canvas are not separated but joined by his initially smudged borders, which provide the unifying effect … Stella's smudging fuses the component lines on first glimpse, never allowing them an independence that would challenge the nonrelational, unitary effect he sought" (Strickland, 2000, 103).

Figure 8: Frank Stella: Zambezi (1959). San Francisco MOMA.

The visual spareness and detachment, but foremost the repetitive elements of his paintings might have been influenced by Samuel Beckett, as Stella pointed out in an interview: „pretty lean … also slightly repetitive … I don't know why it struck me that bands, repeated bands, would be somewhat more like a Beckett-like situation than, say, a big blank canvas" (Antonio, 1981, 141).

After the black paintings followed a series of ‚white' or ‚aluminum' or ‚notched' paintings; here again, although following Rauschenberg in color, the approach was radically different: rather that drawing patterns within the classic rectangular format, he cut out those parts of the canvas that did not conform to the pattern. This series is drawn only with verticals, the texture of the aluminum paint is smoother and the use of penciled guide-

lines creates an impersonal regularity in the pattern – often seen as a precursor to the industrial streamlining of Minimalist sculpture. Stella became the principal figure in the new attitude of American ‚directness' and ‚anti-illusion'—he famously summed this up in an interview with Glaser (1966, 158) as „What you see is what you see"—only what can be seen is there. However, he was not alone in proposing this new sense of wholeness, with both Newman and Pollock preceding him; as Meyer (2001, 90) notes, „his painting-reliefs heightened an awareness of Rauschenberg and Johns … The relationships were complex: the impact of Stella's work … elicited quite different readings."

The uncompromising two-dimensionality of Stella could be seen as the end-product of an evolution of abstraction that had been started with the Cubists who had freed themselves from the window of perspective: „In Stella, the surface was neither a window into an illusionistic world nor a skin for tattooing, but everything, all there was. It is more than a facile oxymoron to note that the (anti-)spatial evolution of abstraction culminated in the concreteness of the objectpainting" (Strickland, 2000, 108). An important characteristic of Minimal Art, namely the development of the objecthood of art works, could be counted among Stella's achievements; although the term had been used before by e.g. Rauschenberg (Colpitt, 1990, 109), Stella reversed the Duchampian notion of object-as-artwork into the artwork-as-object.

2.1.4 Radical Minimalism and Post-Minimalist Painting

Later minimalist painters further radicalized reduction, a tendency that might have been aided by the need to distance themselves from their predecessors. A most notable example was Robert Mangold: „He has excluded from his work all such concerns as illusion, image, space, composition, climax, hierarchy of interest, movement, emotional content, painterliness, interest in materials or processes, and any sort of association or reference to anything other than the physical painting itself … To have produced work of intellectual and visual power with such severity of means is impressive, and he [Mangold] is certainly among the most important of the ‚Minimal' artists" (Spector, 1974).

Figure 9: Robert Mangold: Four Triangles Within a Square (1974). Dallas Museum of Art.

Robert Ryman, known primarily for his ‚achromatic' paintings, created in experiments dating from 1958 and executed since 1965, was one of the few painters able to follow Stella with a distinctive style. His ‚white' paintings are less rigidly white, the bands often interact with the different colors of his supports, normally are contiguous, and often expressively brushed, without an attempt of art concealing art. This painterliness – although the bands are still parallel to the frame, the furrows of the brushwork „suggest[s] fine gradients of linearity" (Strickland, 1991, 110) – is in stark contrast to the bluntness of Stella's works. Still, the explicit brushwork might be interpreted as a continuation of the reductivist aesthetic of economy and exposure of means, as Ryman stated: „There is never a question of what to paint, but only how to paint" (Art in Process at Finch College catalogue 1969, quoted in ibid., 110).

Richard Tuttle, a less prominent Minimalist figure, might have been the painter who expressed the objecthood of the artwork in most radical terms. Like Stella, he was rather young when he had his first show at twenty-four in 1965 at the Betty Parsons Gallery. He is known for the modest nature of his work: in the seventies, his works consisted of an inch of rope or a foot of polygonized wire nailed to the wall. Although these works count among the least flattering in art history, they were judged to be „not beautiful. But … in some sense outrageously ‚poetic'" (Perreault, 1968, 17).

A continuing refinement of minimalism can be observed in the later works of minimalist painters. As the most drastic, or primitive interpretation of minimalism was already taken by Rauschenberg, more subtle forms of minimalism evolved. The question of defining a minimalist aesthetics emerged: What is a great work? How is art determined? The influential art critic Clement Greenberg defined artistic ‚quality' e.g. for the zip paintings of Newman that „look easy to copy, and maybe they really are. But they are far from easy to conceive" (Greenberg, 1962). Judd argued that a new kind of painting had been developed: „In earlier art the complexity was displayed and built the quality. In recent painting the complexity was in the format and the few main shapes … A painting by Newman is finally no simpler than one by Cézanne" (Judd, 1965, 184). Judd used ‚interesting' as a criterion to determine works that are not ‚merely interesting' but ‚worth looking at'. While „Greenberg is the critic of taste …... behind every one of his decisions there is an aesthetic judgement", „in the philosophic tabula rasa of art, »if someone calls it art,« as Don Judd has said, »it's art«" (Kosuth, 1991, 17).

2.1.5 Judd, Andre, Flavin and Morris: Minimal Objects

It looks like painting is finished
– Donald Judd

The role of sculpture in minimalist art is subject to scholarly debate. While for many artists and critics, sculpture provided new opportunities, after all possible reductions had been tried in painting, for others, the third dimension does not add significant new elements. Strickland notes: „the three-dimensional art is fundamentally an outgrowth of earlier work in its sister medium" (Strickland, 2000, 259).

Donald Judd, originally a painter, turned to sculpture as he disliked the „‚illusionistic quality of painting'" and insisted on the necessity of working in „‚real space'"; the very development that minimal painting took, further and further reducing composition and painterliness made it difficult to continue further within the same medium. In sculpture as well as in painting, the question about how far reduction could go was soon raised as technical bravura and richness in composition or color lost their importance. By the end of 1964, Judd no longer created his sculptures by himself, but rather „sent out plans for works to factories, whose superior execution, he felt, would make his work perspicuous" (Meyer, 2001, 81). As the new art approached the asymptotic limit of the readymade and the bare canvas, it became ‚concept art'; yet, at the same time, this opened the doors for criticism—Kramer suggested that this art was too easy to reproduce to be called art, „the work wasn't crafted enough … too simple to look at … boring" (ibid., 81).

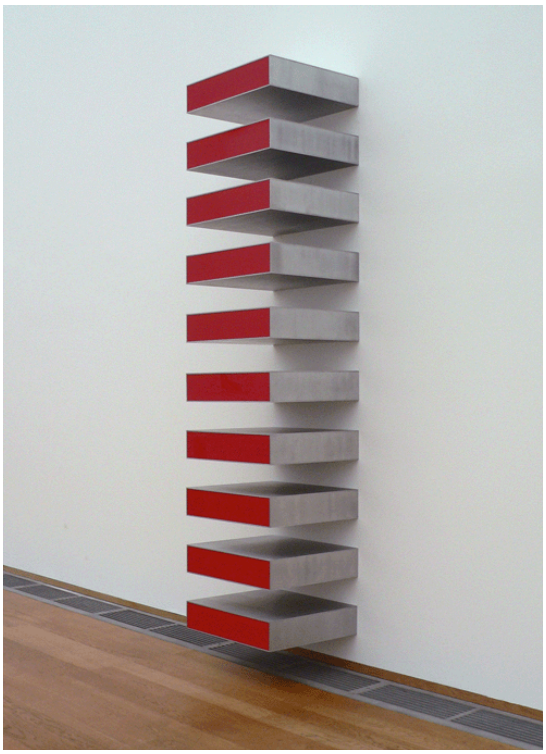


Figure 10: Donald Judd: Untitled (1976).

In a discussion with Frank Stella, who was named as a principal influence on the object-sculptors, Donald Judd, one of the most prominent of minimalist sculptors, identified himself as seeking for wholeness in painting. This was the reason for symmetry in his work as he „wanted to get rid of any compositional effects, and the obvious way to do it was to be symmetrical"[1] (Glaser, 1966). Naming Newman as an example, he insisted that „You should have a definite *whole* and maybe no parts, or very few … The whole's it. The big problem is to maintain the sense of the whole thing" (ibid.). As Judd and Stella both

1. Judd later elaborated this position in „Symmetry" (1985, reprinted in Judd, 1975, 92-95)

knew nothing of similar tendencies in European art (Meyer, 2001, 88), they identified this objection to relational composition as a major feature of the typical American minimalism. An important aspect of minimal sculpture, is thus the wholeness of Gestalt; artists argued against the visual separation of parts in their specific objects. Retrospectively, wholeness became a key quality of both painting and sculpture (ibid., 134ff).

In the same interview, Judd places a strong verdict on previous approaches to minimalism; for him, „painting is finished" (Glaser, 1966). At the same time, he insists that in form, „the new work obviously resembles sculpture more than it does painting, but it is nearer painting" (Judd, 1975, 183), and coins the term *specific objects* for Morris' and his own work.

A prototypical work that was subjected to the accusation of trivial simplicity, and thus a lack of artfulness, was Andre's Lever (1966) consisting of 137 aligned firebricks (Figure 11); a satire ‚exhibition' at Chapman College in Orange, California, organized by Harold Gregor, director of the college's Purcell gallery stated that anyone „could purchase a similar set of bricks and make an identical work", even „the subtraction or addition of a few dozen bricks" would „in no way change its form ….. [that was] boringly minimal in content" (quoted in Meyer, 2001, 82). However, this critique too simply dismissed the Lever's form as visually dull, and disregarded Andre's intention of demonstrating the nature of the building blocks in his art that was „installed by Andre himself, who prized each brick for its own material sake; and understood by the artist to be perfectly continuous with the modernist tradition of formal innovation rather than a dadaist legacy" (ibid., 82).

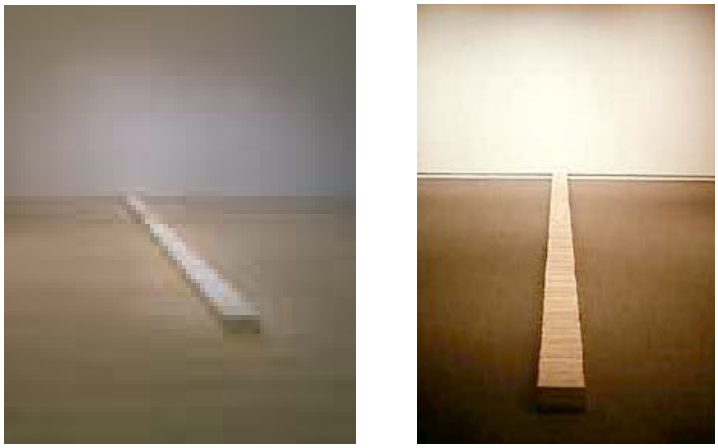


Figure 11: Carl Andre: Lever (1966).

The object character of minimal sculpture sharply contrasted even the most minimal painting. As one of the most radical protagonists of minimal sculpture, Dan Flavin used fluorescent light tubes for his works—while others like Judd developed his works out of industrial materials, he chose industrial objects as his medium: „my work becomes more and more an industrial object the way I accept fluorescent light for itself" (ibid., 92). Painting expert Bob Rosenblum noted that Flavin had thus „destroyed painting" for him (Judd, 1965, 189).

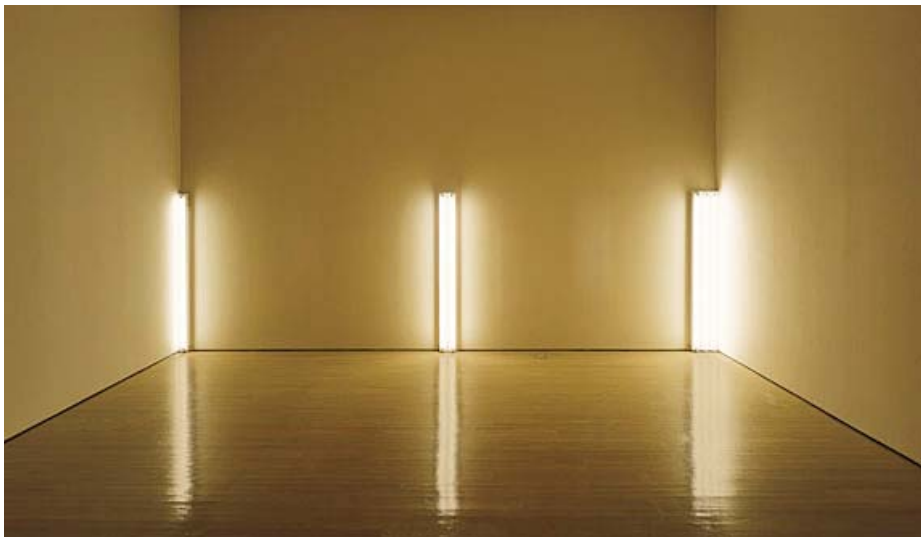


Figure 12: Dan Flavin: The Nominal Three (to William of Ockham), 1963.

Robert Morris introduced an art that—as architecture—related to human scale, „Architecture, the body, movement – these are the terms of Morris's early minimalism" (Meyer, 2001, 51). While his early artworks tried to interface with the context—as has been noted in different sources, a sculpture called ‚Column' was used as a prop in a performance at La Monte Young's Living Theater in New York in 1962 (Krauss, 1977)—he later removed himself from ‚such allusions' and argued for a purely abstract art (Morris, 1966) before returning to architecture as a source of inspiration in the 1970s. Yet, even in his abstract period, he produced an art that was quite different from Judd's plainly pictorial model: While Judd's „Specific Objects' were there to „look at', Morris' works „„were to be experienced by an ambient body that walked around, and through, the work itself" (Meyer, 2001, 51).

Figure 13: Robert Morris: Corner piece (1964) Dallas museum of art.

Returning to the question of wholeness, for Morris—who understood the term primarily in perceptual terms—only sculpture could be whole, a quality he referred to as Gestalt, as „wholeness seen'. Judd, who started out as a painter rather than as a performer, admitted he did not want to „consider the viewer ..... I'm rather interested in what I want to think about, what I want to do." (Glaser, 1966). While Judd carefully built and exhibited his works so that they could be seen, Morris consciously theorized the experience of his art by recourse to ultimate reduction, and proposed that „the most reduced shapes were the most desirable" (Meyer, 2001, 159). For his use of the pictorial relief, Morris even accused Judd of illusionism, „the autonomous and literal nature of sculpture demands that it have its own, equally literal space – not a surface shared with painting." (Morris, 1966), and claimed that „wholeness was an integral quality of sculpture alone, for wholeness could only be seen in three dimensions" (Meyer, 2001, 158).

In contrast with Judd's pictorial conception of artistic sculpture, the Specific Object that „presents a static, articulated shape to a viewer's gaze" (ibid., 51), Morris' involvement in performance art included the architecture, the human body, and movement in his understanding of sculpture. Judd was the leading advocate of the position Fried characterized as »literalist« (Fried, 1966, 22) – the compulsion to rid art of illusion, which, resulting in the production of objects, rendered painting obsolete (Fried, 1998, 12)" (Meyer, 2001, 230), while „Robert Morris conceives his own unmistakably literalist work as resuming the lapsed tradition of constructivist sculpture" (Fried, 1998, 12)

2.1.6 LeWitt: Minimal Structure in Minimalist Sculpture

Sol LeWitt, a graphic designer who worked for I.M. Pei in the mid-fifties, contributed his focus on structure to minimalism: „his major achievement in the 1960s as the assorted cubical structures whose skeletal nature was reinforced by their whiteness" (Strickland, 2000, 271). Echoing the five-foot square format of Reinhardt's paintings, he built his works based on five-feet-cubed cubes. By exposing the structure of his works, he created a „sense of dematerialization akin to the experience of both X-rays and architectural plans showing us the fragile framework … which underlies the facade of daily existence" (ibid., 271).

Beginning with his second exhibition at the Dwan Gallery in New York in 1966, LeWitt experimented with white, modular open structures. He chose White as he considered it to be „the least expressive color" (quoted in Cummings, 1974)—white „enhanced the reading of his delicate geometries" (Meyer, 2001, 200). Feeling limited by the literalist tendencies of other minimalists, and having the impresson that „reductive Minimalism was self-defeating" (ibid., 202), LeWitt used repetition to further expose not only the internal structure of an artwork, but also the system behind the internal structure. While the minimal object repressed its conceptual aspect, for LeWitt geometry could become „a machine that makes the art" (LeWitt, 1978a). He chose the cube at it itself „is relatively uninteresting … Therefore, it is the best form to use as a basic unit for any more elaborate function, the grammatical device from which the work may proceed" (LeWitt, 1978b).

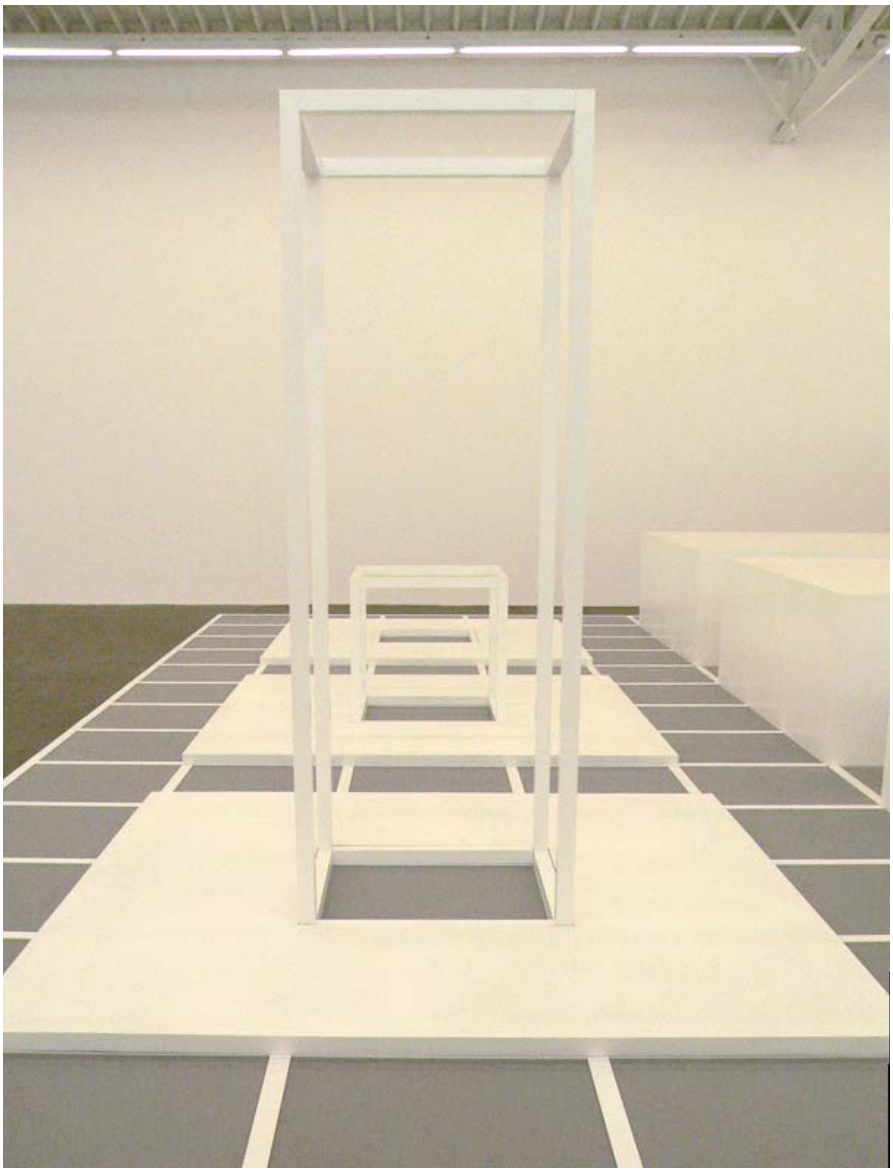


Figure 14: Sol LeWitt: Serial Project #1 (Set C), 1966/85. Flick Collection Berlin.

2.1.7 Post-Minimalist Sculpture

Although minimalism itself is often seen as being limited to the 1960s (comp. Strickland, 2000, 6f), it formed an impression of art that was to last longer. Richard Serra, like Judd originally a painter, ranks among the most popular post-minimal artists. In his massive, stark works, optical qualities of minimalism are mirrored. However, most of his works „have an inherently more kinetic and menacing aura than the static and indifferent quality of earlier Minimal sculpture" (ibid., 290). The monumental nature of most of Serra's later works tends to ignore the spectators, or at least make them feel insignificant; in his *Stacks*, massive sculptures „do not threaten to collapse on the spectator, enclose him in a vise, or even block his exit. They merely face each other and ignore him." (ibid., 291)

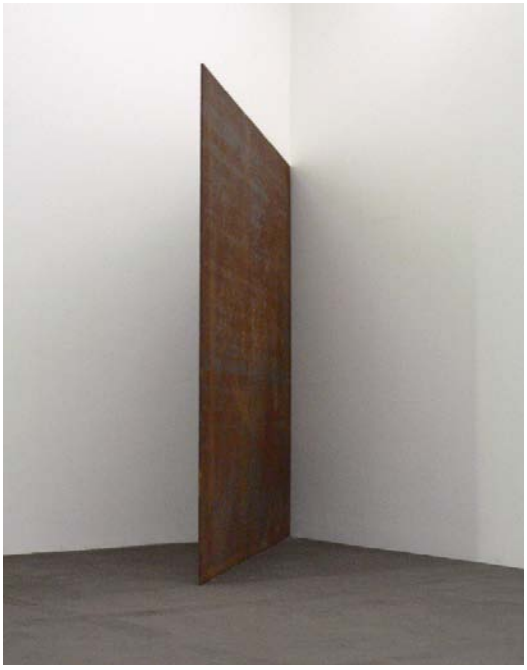


Figure 15: Richard Serra: Strike (1970). Flick Collection, Berlin.

Other artists took different, and more inviting approaches to interacting with the audience. The non-communication of spectator and artwork was inspiration e.g. for Bruce Nauman's Sealed room. In this sculpture, a room—four perfectly blank, white walls— is both presented to and hidden from the visitor. There is no way to determine what the sculpture looks like from the inside, and to add to this suspense, a humming voice is created by several large fans inside the sculpture. The artwork presents no image, and thus becomes a projection wall for the spectator's imagination.

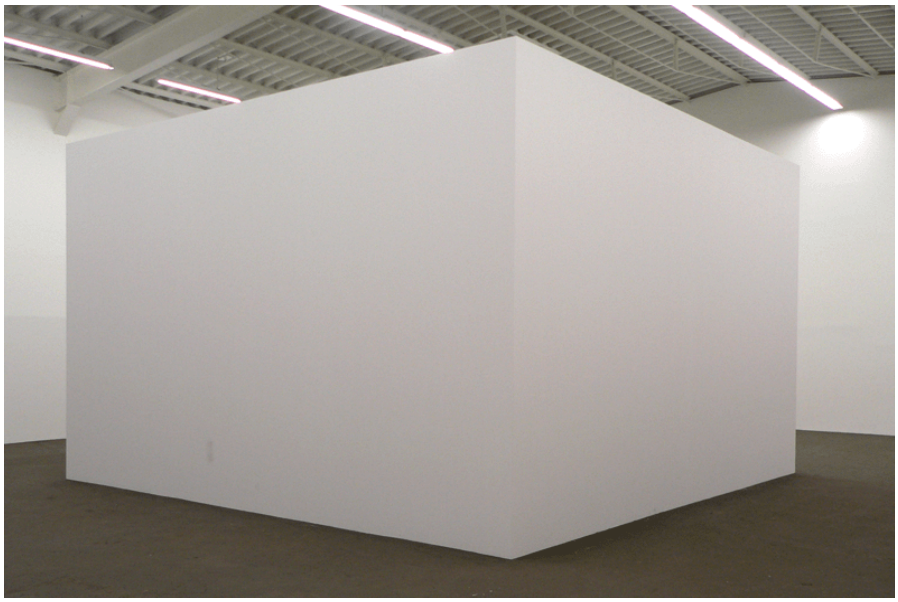


Figure 16: Bruce Nauman: Sealed room – no access (1970). Flick collection, Berlin.

2.1.8 Summarizing Minimal Art: Art as Art or Cooperative Sense-Building?

For the listener, who listens in the snow,
And, nothing himself, beholds
Nothing that is not there and the nothing that is.
– Wallace Stevens, The Snow Man

Although the first impression of a work of Minimal Art may be bewildering, it is also intriguing; it generates a need to explore, to solve the ‚puzzle', to fill in the details missing in the framework. This immediate interactivity that most strongly surfaces in the monochrome paintings, also of Reinhardt, is contrasted with his dictum of ‚art as art'. Ultimate success in not only granting raw material its independence, but in insisting on conserving its modality – Stella trying to keep the paint „as good as it was in the can" – would create an art-object that simulated complete independence from the human will that created it. Stella objected to viewers „who … retain the old values in painting … If you pin them down, they always end up asserting that there is something there beside the paint on the canvas" (Battcock, 1968, 157f.).

Consequently, some minimalists tried to eliminate the viewer from their art. If this degree of disconnection to anything human were reached, little value for the viewer could be found. Critic Naomi Spector enthusiastically hailed Mangold for excluding „from his work all such concerns as illusion, image, space, composition, climax, hierarchy of interst, movement, emotional content, painterliness, interest in material or processes, and any sort of association or reference to anything other than the physical painting itself." (Spector, 1974) This perspective was not widely shared, and Strickland attributes her interpretation to „the Me Decade, an artistic analogue of Self magazine (I have no concern for the city, country, planet, or universe, but I'm looking good)" (Strickland, 2000, 114).

Figure 17: Richard Serra: Tilted Arc (1981) on the Federal Plaza in New York.

The creation of entirely self-sufficient objects without a relationship to a bigger whole would deny any possibility of interaction. Only few minimalists have tried to thus dom-

inate the viewer; perhaps fortunately, Serra was unable to realize his statement about public places: „one has to consider the traffic flow, but not necessarily worry about the indigenous community, and get caught up in the politics of the site. There are a lot of ways one could complicate the problem for oneself. I'm not going to concern myself with what ‚they' consider to be adequate, appropriate solutions" (Serra, & Weyergraf-Serra, 1980, 63). After he had installed his metal arc in 1981 (Figure 17), the public reaction was such that he was „perforce very much concerned with such issues … he had blocked the view and escape of residents and employees on his sites. This time it backfired: the work was dismantled beginning March 15, 1989 after extensive and controversial public hearings" (Strickland, 2000, 290).

Although some artists either ignored the audience, or tried to exactly define the nature of perception for their art, most minimalist artists actively involved the context, and the viewer in the interpretation of their artworks. Elayne Varian (1967) introduced a minimalist show highlighting the relationship of viewer and object: „The purpose of this exhibition is to show the attitude of contemporary sculptors to scale and enspheric space … It is possible to have positive (enclosure) and negative (exclusion) attitudes to define space, … equally important [is] their approach to the surrounding space, the negative space". And critic Richard Wollheim (1965) notes that especially in the reductionist artworks „the canvas, the bit of stone or bronze, some particular sheet of paper scored like this or that" concentrates attention upon individual bits of the world.

Minimalist artists focused on a constructive stance in the interaction of viewer and art work; they were not interested in artfulness, and consequently they disliked artful analyses of their works. What was more important was the immediate effect of a work: „One could stand in front of any Abstract-Expressionist work for a long time, and walk back and forth, and inspect … all the painterly brushwork for hours. … I wouldn't ask anyone to do that in front of my paintings. To go further, I would like to prohibit them from doing that … If you have some feeling about either color or direction or line or something, I think you can state it. You don't have to knead the material and grind it up. That seems destructive to me; it makes me very nervous. I want to find an attitude basically constructive rather than destructive" (Glaser, 1966, 159).

2.2 Minimalism in Music

As the term minimalism was first used to describe painting, adoption is part of the character of all subsequent uses. Bernard (1993) describes how critics who argue that the term may hardly apply to music, who dispute „the extreme reduction of the musical means is important enough to function as a fundamental characteristic of this music„ (Mertens, 1983, 11-12), oversee that the term minimal—as the last part tried to illustrate—goes beyond mere reduction in meaning. It has also been quite successful, as other labels, such as „phase-shifting", „repetitive", „systemic", or „process" music, are no longer used to refer to what is now known as *minimal music*. Manyfold connections of painters and musicians can be observed in the New York art scene in the 1960s. The earliest performances of

minimal music took place in art galleries, and individual figures collaborated over extended periods of time: Philip Glass with Richard Serra, or La Monte Young with Robert Morris, to name just a few. Bernard notes that „it is interesting that no one seems to have thought to apple the term *minimal* to this music, at least in print, before Michael Nyman did it sometime in the early 1970s" (Bernard, 1993, 87).

In the following sections, the ‚minimalist' in minimal music is retraced to use the many similarities with minimal art and some of the differences that emerged as minimal music took a more independent path to add and differentiate the previously detailed meanings of minimalism.

2.2.1 The Origins of Minimal Music

> I have spent many pleasant hours in the woods conducting performances of my silent piece…for an audience of myself, since they were much longer than the popular length which I have published. At one performance…the second movement was extremely dramatic, beginning with the sounds of a buck and a doe leaping up to within ten feet of my rocky podium."
> — John Cage, Silence: Lectures and Writings

After studying classical avant-garde with Arnold Schönberg in Los Angeles in 1934, John Cage became one of the main protagonists of ‚Concept Music', a style of music where the composition dominated the performance. He conceived of pieces where tossing coins determined musical events (*Music of Changes*, 1951), wrote for 12 radios (*Imaginary Landscape no.4*, 1951) and composed what was seen by some as „the prototype, the primordial piece of Minimalist conceptual art" (Rockwell, 1986), a piece called *4'33"* (1952) where the performer is directed to do nothing at all for that length of time. Strickland suggests that this piece was inspired by Rauschenberg's all-whites and represents the „conceptual ne plus ultra for radical reductivism" (2000, 30).

Concept music, with the Fluxus group being the most important source, is often considered as the predecessor of minimal music; Nyman describes that the Fluxus composers „reviewed multiplicity, found its deficiencies, and chose to reduce their focus of attention to singularity" (1972,119). Yet, concept music often seems to border on arbitrariness—the limits composers set to chance are few (Bernard, 1993, 95), and it is difficult to decide whether only irony, or a serious intention (as is assumed in minimalist art) drives its principals. This question of control marks the differences between concept music (that widened the definition of what can be considered music) and early minimal music, which implemented „formal control" of chance elements (Gena, 1981).

Cage (1991) hints at nihilist Dada as a source of inspiration. This emphasis on the illogical and absurd, the creative techniques of accident and chance demonstrates the difference between Cage and some of the early pieces of La Monte Young and what later developed into minimalism. Bernard (1993,96) points out, that – like Minimalist artists (e.g. Morris, Flavin, or LeWitt) deal with temporality using serialism, or a gradual or system-

atic progress through a series of possibilities – Minimalist composers emphasize the passage of time by „*composing out*" the possibilities of their material *in a transparent manner*. Minimalism is thus marked by a conscious and transparent for of reduction that uses omission as an explicit element to target attention.

Perhaps even more important for the minimalist composers themselves, minimalism in music is both a counter-reaction and a derivation of the avant-garde style of serialism (Potter, 2000, 20), and can also be understood in political, rather than aesthetic terms. Serialism had long been the predominant form of „new classical" music, dictating what defined good music, and even determining which techniques to employ when composing; it had become very difficult to think of a new form of music as the atonal revolution of „old" classical harmonics and structural principles had created new, equally restrictive rules. Minimalism established a radically different alternative both to the „old" and „new" classical music. For David Lang, minimalism „was a historic reaction to a sort of music which had a strangle hold on … American musical institutions, and which none of us really liked. … What most people really hated was the way that this other world had theorized that it was the only music possible … I look at Minimalism … as being just the battleground that was necessary to remove those forces from power: not to obliterate them or destroy them, but … to loosen up the power structure in America. And I think that [one reason why] Glass's music and Reichs's music came out so severe, and so pared down, was that … it was a polemical slap in the face … That battle's been fought … My job is to sift among the ashes and rebuild something" (Lang quoted in ibid.).

The suggested lineage of minimalism includes also some more distant predecessors, yet in Richard Wagner's *Das Rheingold* (1854) the simplicity of long chords in the opening bars rather amounts to anticipation of the complex, virtous tapestry that follows, Eric Satie's *Vexations* (1963) simply employs repetition without any development as an artistic means, and the sometimes mentioned *Bolero* by Maurice Ravel (1928) might be repetitious, yet the „Romanticism of the piece is the antithesis of austere Minimalism" (Strickland, 2000, 124) and its directionality in dynamics and rhythm is unidirectional.

Concentrating on the four musical figures most prominently identified as ‚Musical Minimalists', Terry Riley, La Monte Young and Philipp Glass are introduced in some detail in the following sections. For a more thorough and balanced account of their lives and works I must refer the reader to Potter's excellent ‚Four musical minimalists' (Potter, 2000) and, for a different perspective, to Strickland's analysis of the origins of minimalism (Strickland, 2000). For a contemporary perspective, the collected articles of Tom Johnson are a worthwhile read. (Johnson, 1991)

2.2.2 Terry Riley

Terry Riley was the minimalist composer with most connections to Cage's concept music, having practiced that style during his early career. Together with Young, however, he began to conceive a new style that would eventually turn into minimalism. A strong European influence, both from his studies and from living in France, links his works to Serial-

ism. Strickland (2000, 133) highlights Riley's „two major contributions to early Minimalism: the reintroduction of tonality" and „the use of repeating musical modules". The former is first introduced in his 1960 *String Quartet*, where he returned to tonality, which had been absent from his earlier works, in long tone composition—fairly unusually long tones, for that matter. The latter became evident in his 1961 *String Trio*: „the unvaried repetition of tonal phrases marked a radical compositional simplification" (ibid., 143). Composing the Trio, his interest in Serialism remained visible—he had studied Stockhausen with Wendall Otey and Robert Erickson at San Francisco (ibid., 133), but tonal repetition became musically dominant, while chromaticism moved to the background; for Riley, „That was the transitional piece" (Strickland, 1991, 112).

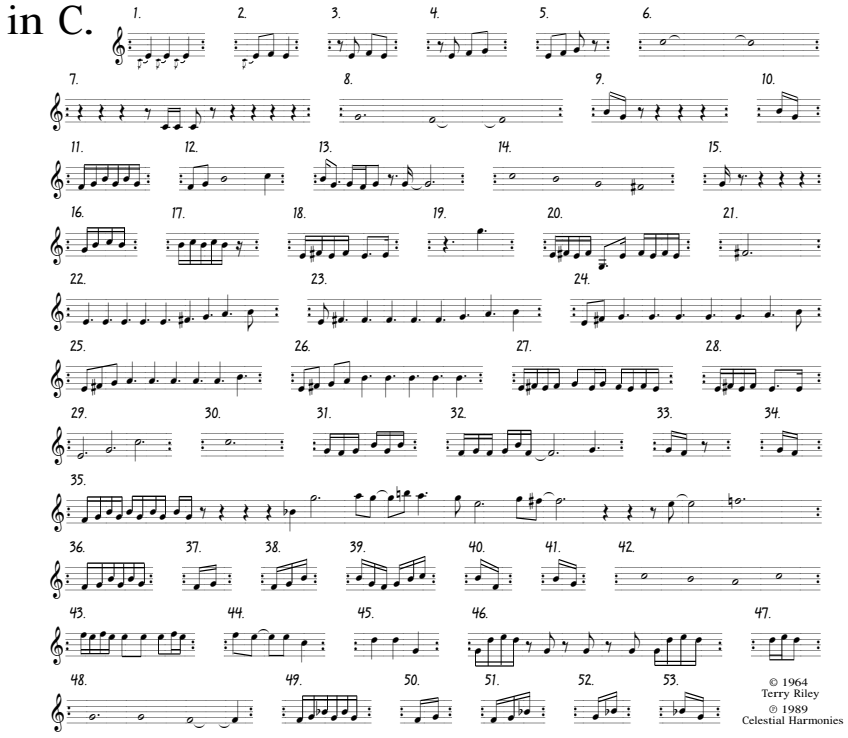


Figure 18: Terry Riley: In C (1964).

Yet, his most important contribution to minimalism was *In C*, fifty-three „modules", supposedly written in one night (all patterns are shown in Figure 18); *In C* is scored for „[a]ny number of any kind of instruments" (Potter, 2000, 112). Its „ensemble can be aided" by a pulse that can be performed „on the high c's of the piano or on a small mallet instrument" (ibid.). Not all performers need to play all the modules; some are better suited to melodical instruments, such as wind instruments, while the faster patterns are fitting to be played by keyboard instruments. The piece starts with a pulse, synchronizing all performers; after that, each musician may start with module one in her own time, repeating as often as wishes before moving on to the next module. Although in some performances, this has resulted in „a kind of glorious, hippie free-for-all" (ibid.), performances produced in cooperation with Riley „generally deployed a maximum of four modules at any time

and thus encouraged the performers to work more closely together" (ibid.); in Riley's words, no performer should draw attention to his own part at the expense of others, this would‚collapse the structure of the piece' (ibid.).

By using repetition and relying on skillful improvisation, the score of *In C* fits onto a single page, and with written instructions is still shorter than three pages, although performances „normally average between 45 minutes and an hour and a half" (Riley, 1964). The resulting patterns are different in every performance, yet the overall impression is not completely different—one can recognize *In C*. This balance between freedom for the performers and careful composition for possibilities that are limited by cues rather than rules: Riley states e.g. „as the performance progresses, performers should stay within 2 or 3 patterns of each other. It is important not to race too far ahead or to lag too far behind." (ibid.) Also, a strict sense of rhythm is demanded for by Riley, to prevent an impression of disorder.

Repetition had been discovered long before by Riley: when he had worked at the San Francisco Tape Music center, he experimented with an echo effect to put a piece of Ramon Sender on a „sonic »acid trip«" (Strickland, 2000, 148). Later, working in Paris, where he played jazz and ragtime in Pigalle, he composed *Mescalin Mix*, a piece elaborating this echo effect. After continuing playing lounge piano in officers' clubs and bars at American bases, he met a French engineer who „ended up hooking two tape-recorders together" (Strickland, 1991, 112). The tape ran along the record head of the first recorder and the play head of the second recorder, who thus played what had ben recorded before, feeding the output back to the first recorder. Effectively, this resulted in a progressively complex structure as the recording was overlaid on itself. What might seem trivial today in an age of sampling was a revolution then, and for Riley a revelation that he was still enthusiastic about 25 years later (Strickland, 2000, 149).

2.2.3 La Monte Young

The music Terry Riley was using for his first tape experiments, *The Gift*, was the piece *So What* from *Kind of Blue*, one of the best-known jazz albums of all time, recorded in 1959 by Miles Davis, who pioneered modal jazz. The scarcity of melodic shifts—*So What* contained only two modes in thirty-two bars—allowed Riley to use repetition without immediately creating disharmonies. Jazz can be seen as one of the major influences on minimal music, Riley, La Monte Young and Steve Reich all played in jazz groups in their teens, and Coltrane, who continued to explore exotic modes „from Morocco to Afghanistan" (ibid., 150), made popular the soprano saxophone that Young adopted for his latest work period, where he turned to drone sounds.

In his early years, Young took up „the indeterminacy practiced by [John] Cage and his followers" (ibid., 144) in his „concept art" composition of April 1960, *arabic numeral (any integer) to Henry Flint*: the arabic number indicated the number of times a sound was to be repeated, with both the choice of number and sound left to the performer. Cage himself found the piece „relevatory" (Cage, 1961, 52). In an extended realization of repetition, Young composed *1698* in 1961, the title designating the number of times the same disso-

nant chord needed to be played on a piano—sometimes, Young was reportedly playing the piano until his fingers bled (Strickland, 2000,145).

Young thus presented himself as uncompromisingly obsessed with repeating sounds—according to Potter (2000, 43,12) he was „wildly interested in repetition, because [he thinks] it demonstrates control". His *Composition 1961* with 29 identical instructions to draw a straight line and follow it represent another form of repetition. Young's repetition differed from the repetition used by Riley in his „still traditional, if highly experimental, score[s] primarily by means of notated and repeated musical phrases" (Strickland, 2000, 145) and has often been connected to a continuity derived from repeated repetitions: Smith (Smith, 1977, 4) notes that „witnessing a single activity extended in time, we begin to appreciate aspects and ideas that would otherwise remain hidden", and Mertens judges: „the term repetition, however, can hardly be used for La Mnte Young's music, since in this case the principle of continuity is decisive." (Mertens, 1983, 16).

What was later termed *additive processes* was—in a simple form—also pioneered by Young, who wrote *Death Chant* for the funeral of an aquintance's child. To a minimal motive of two notes, a single note is added for every repetition; the resulting piece thus, although repetitive, demonstrates the slow progression of time (Figure 19).

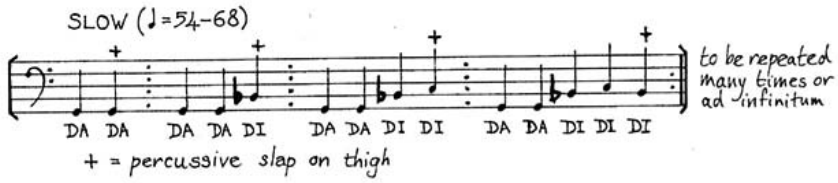


Figure 19: La Monte Young: Death Chant (1961).

John Coltrane, who „used to construct modes or sets of fixed frequencies upon which he performed endlessly beautiful permutations" (Young, & Zazeela, 1969), influenced Young to begin playing the saxophone, and partly by that instrument, a new composition phase was introduced: As in the 1964 *Sunday Morning Blues*, rapid five-second bursts of notes on Young's sopranino saxophone were contrasted with voice, guitar, and viola drone sounds that were held continuously through the performance (Figure 20).

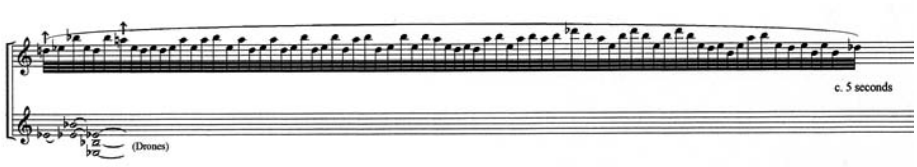


Figure 20: Sunday Morning Blues (1964).

As Young himself notes (Young, 2001), the „use of sustenance became one of the basic principles of my work. When there are long sustained tones, it is possible to better isolate and listen to the harmonics." This principle has been perfected and applied in Young's best-known piece, the 1958 *Trio for Strings*, „which, while constructed as a serial piece, has pitches of longer duration and greater emphasis on harmony to the exclusion of almost any semblance of what had been generally known as melody." (ibid.) The Trio takes serial

notions such as symmetrical row construction and static tonal surfaceto an extreme conclusion, and is comprised only of long sustained tones in varying alignments alternating with silences (Figure 21).

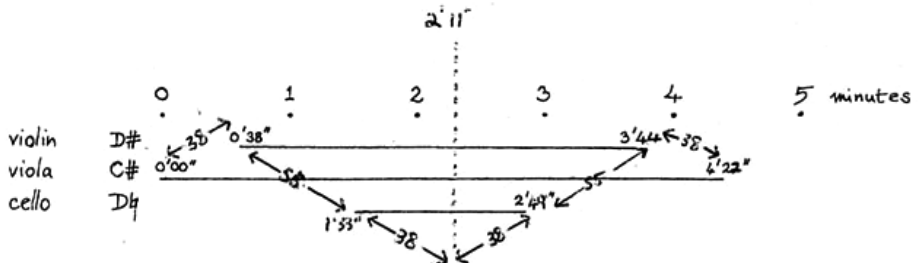


Figure 21: Trio for Strings (1958).

The use of long-held tones eventually led Young to the development of his „concept of the drone-state-of-mind" (ibid.); behind the use of sustained drone sounds stands the belief that „tuning is a function of time"(Young, & Zazeela, 1969), and that „frequency environments" influence the nervous system, eventually „establishing periodic patterns" (Young, 2001). Young later became a disciple of the Pakistan (then-Indian) Raga singer Pandit Pran Nath, whom he admired for his ability to perfectly hit and sustain pitch (Strickland, 2000, 172f).

2.2.4 Philip Glass

Philipp Glass' interest in reduction developed under the influence of Samuel Beckett (Potter, 2000, 255): during his Paris studies, he created his first theatre piece together with JoAnne Akalaitis. This play was later described by Glass as „a piece of music based on two lines, each played by a soprano saxophone, having only two notes so that each line represented an alternating, pulsing interval. When combined, these two intervals (they were written in two different repeating rhythms) formed a shifting pattern of sounds that stayed within the four pitches of the two intervals. The result was a very static piece that was still full of rhythmic variety." (quoted in ibid., 256)

A second important source of inspiration was Indian classic music (ibid., 257ff), whose rhythmic qualities fascinated him: he felt that in contrast with the western ‚divisive' rhythm that divided a musical score in bars, and these again in beats, Indian (and other non-western) music created rhythms using additive methods (ibid., 270f). These influences lead him directly to his contribution to minimalist music, additive music became the „structural essence" (ibid., 284) of his musical work.

1+1 (1968), a piece for one player and amplified tabletop, is often taken as the initial milestone in Philip Glass' development. It is the earliest and most rigorous example of the composers use of what he termed an ‚additive process'. *1+1* is only concerned with rhythm: the player taps on a tabletop amplified via a contact microphone, two basic ‚rythmic units' are offered and amended by suggestions how to use them as ‚building blocks' for a performance. The ‚additive process' is specified simply by instructing the player to combine the „two units in continuous, regular, arithmetic progressions"; the first of Glass'

examples comes out as „1 + 2; 1 + 2 + 2; 1 + 2 + 2 + 2; 1 + 2 + 2; 1 + 2 etc.“: the first unit is contained only once in each alteration while the second unit expands and contracts symmetrically (Figure 22).

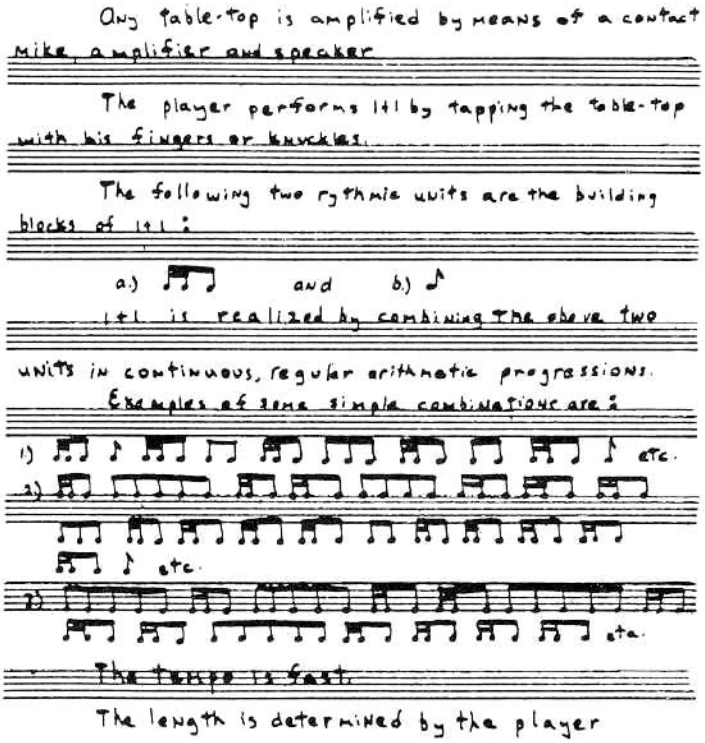


Figure 22: Philip Glass: 1+1 (1968).

This „additive process“ served Glass as the main technique for structuring his compositions for some ten years from thereon. He remarked: „It's funny, it's such a simple idea, but believe it or not, I just hadn't thought of it then. Actually, it was the result of a year or two year's work: I looked back and thought of simplifying all the processes I had used into that one idea“ (quoted in Potter & Smith, 1976).

Although Glass had previously applied similar techniques to sequences in his music, *1+1* is the first work that rigorously regimented the technique, applying systematic rules for expansion and contraction and thus making the unfolding structure clearly audible, transparent. As with Reich's works using phasing, „the compositional process and the sounding music become one“ (Potter, 2000, 272).

Glass invented additive processes. They are no cyclic structures, and they differ from the repetition that La Monte Young developed from working with tape interference: Glass additive processes follow strict rules, yet they describe a dynamic movement. The rules for creating structure are formalized, and can be perceived by the listener as such, yet the possibilities for combination allow the weaving of complex musical patterns. This also goes far beyond the inspiration that Glass found in Indian music: „The kind of additive processes which Glass made the basis of his own music are not, however, to be found in

Indian practice; even the rigorous application of these is not a direct borrowing but an extrapolation of the composer's own from the Indian approach to rhythm." (ibid., 273).

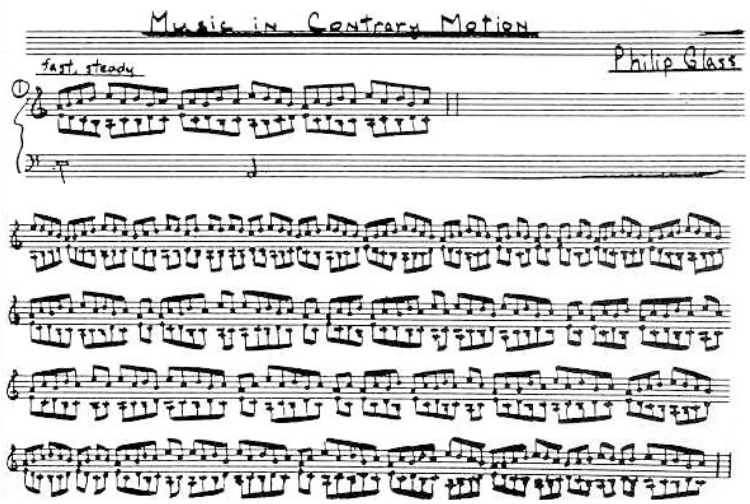


Figure 23: Philip Glass: Music in Contrary Motion (1969).

Philip Glass continued to build more and more complex musical pieces on the principle of additive processes. He experimented with parallel movements in *Music in Fifths* (1969) and *Music in Similar Motion* (1969), and with tonal inversion in *Music in Contrary Motion* (1969) (ibid., 292-300). While the latter was written in ‚open form'—never ending, the piece just stops without reason (Figure 23), he later turned away from such playful experiments and sought to develop additive processes further. In what is considered his masterpiece, *Music in Twelve Parts* (1971-1974), construction rules guide not only rhythm, but also structure: the composition was first played in 1971—and later turned into the first of twelve parts that followed a regular development (ibid., 312f).

2.2.5 Steve Reich

Steve Reich has become the most successful of the minimalist composers in commercial terms, and as his later works are much more compatible with public taste, it has been forgotten that he was among the most rigorous minimalists. Brought up with ‚classical' music—his mother became best known as the Broadway singer June Carroll, his interest quickly turned to both older and more modern music: „It wasn't until the age of 15 that I heard the music that would end up motivating me to become a composer and informing what I did: that was jazz, bach and Strawinsky …" (quoted in Smith, & Smith, 1994, 212). Reich majored in philosophy at Cornell University, and studied subsequently privately with Hall Overton in New York before attending Juillard school between 1958 and 1961—at which time he met fellow student Phillip Glass (Potter, 2000, 155). „Running away from home" (Reich quoted in ibid., 156), he arrived in San Francisco where he composed *Music for String Orchestra* (1961), a twelve-note set that was to be repeated ad infinitum, and *Four Pieces* (1963), „twelve-tone jazz licks trying to become tonal" (Reich quoted in ibid., 159).

Starting with *Four Pieces*, reich felt „despite my limitations as a performer I had to play in all my compositions"—a decision whose practical consequences helped him towards a simpler, reductive style (ibid.). Meeting Terry Riley in 1964, and assisting the development of *In C* (ibid., 111-116), he was „pointed the way towards a more organised and consistent kind of pattern-making with highly reductive means" (ibid., 164). With *It's Gonna Rain* (1965), Reich sought to reproduce the perfect synchronization of tape loops, using a Black American English sermon of Brother Walter; the first movement using only the words ‚it's gonna rain', the second drawing on a longer passage. However, technical imperfection of the Wollensak tape recorders he used let the two recordings fall out of synch with one tape gradually falling ahead or behind the other due to minute differences in the machines and playback speed. Reich decided to exploit this *phase shifting* to explore all possible recursive harmonies.

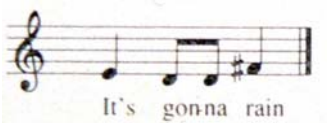


Figure 24: Steve Reich: It's Gonna Rain (1965) – basic unit.

When Reich returned to New York in 1965, he came into contact with minimal artists like Robert Rauschenberg, Sol Le Witt, and Richard Serra. Although he „was derisive about the term »minimalist«" (Reich quoted in ibid., 171), he acknowledged „there certainly was an attitude" (ibid.) he shared with the painters and sculptors; he was also financially supported by the already established Le Witt and Serra as they purchased his original scores. He slowly built up an ensemble that became the center of his compositional activities. The life performance in „painstaking rehearsals" provided the basis for the development of compositions like *Drumming* (1970–71) as they allowed for „many small compositional changes while the work is in progress and at the same time [building] a kind of ensemble solidarity that makes playing together a joy" (ibid., 198). The composition was now only part of the musical product, and the performance and interpretation gained an important role.

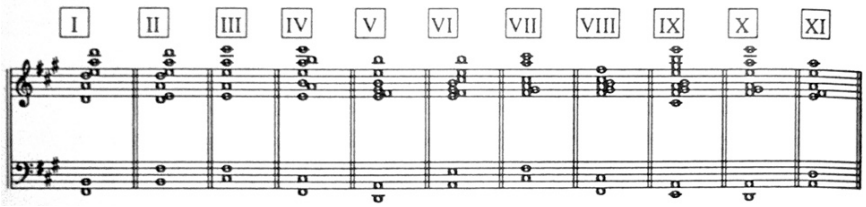


Figure 25: Steve Reich: Music for 18 Musicians (1978) – cycle of chords.

This development culminated in *Music for Eighteen Musicians* (1976): rehearsals for this piece were held over a period of two years, and „structurally, the work is so complex that no attempt was made to notate repeats or achieve synchronisation in what was written down" (ibid., 199). The first complete score was published only in 1996, and consisted of a transcription of a 1978 recording. *Music for Eighteen Musicians* became Reichs' most well-known piece and is valued for its „significant expressive extension of the composer's musi-

cal language" (ibid., 231). Reich still uses the development of patterns and rhythmical shifting as techniques, but combines them more subtly, for the first time exploring the technique of „underpinning a repeated melodic pattern by rhythmically shifting chord changes" (ibid., 233).

Reich gradually developed his interests „from ‚structure' as such to a greater concern with sound" (ibid., 211). In *Drumming*, he introduced a rich variety of instruments—e.g. marimbas and glockenspiels, and combined them with voices (ibid., 212-225). This more symphonic style influenced his later works, although he sometimes returned to a more minimal instrumentation, e.g. in the 1972 *Clapping Music* that was written for the pairs of hands of two performers. Although the choice of ensemble was minimal, *Clapping Music* was still a highly complex piece, its basic unit throwing the 12/8 meter in doubt as notes follow on the 5th, 8th and 10th quaver. The resulting rhythm can be interpreted „in a number of ways; when performed in canon, the choice of downbeats is naturally increased" (ibid., 225). Listening to the piece, it can even be difficult to discern that both players clap the same pattern.

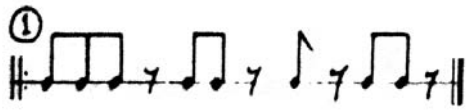


Figure 26: Steve Reich: Clapping Music (1972).

A significant portion of Steve Reich's reputation has been made in the world of pop music, where Brian Eno and Mike Oldfield popularized music influenced by Reich and other minimalists. Reich's influences in pop music can be retraced even in today's music. In the 1999, *Reich: Remixed* was produced, with several leading ‚techno' artists, including Coldcut, Ken Ishii, and DJ Spooky, reinterpreting the composer's music from 35 years.

2.2.6 Summarizing Minimalism in Music

Reviewing the contributions of the four musical minimalists to minimal music, similarities with minimal art are obvious: a reduction in means can be seen in the sparse orchestration of many pieces. Cageian influences lead also to some experiments with the reduction of meaning, yet as a whole, musical minimalism is less playful than its counterpart in art. The main focus of minimal music is on the reduction of structure, and sound—often by employing repetition and through combination of simple patterns.

Distancing himself from Serialism, Riley reintroduced tonality to modern music, and pioneered the use of repeating musical modules. La Monte Young started his composing career with tape experiments, slowly escaping the Cageian influence. His mature minimalist works experiment with repeating sounds, introduce rhythmic additition, and make use of drone sounds to create powerful soundscapes. Philipp Glass was interested in structure and rhythm from his first works under the influence of Beckett, and later perfected structural transparency in his works by inventing additive processes that combine simple patterns using composed rules. This allows both the interpretation to *create* music

based on the composition, and the listener to reconstruct the act of creation. Steve Reich also used reductive patterns and tape loops in his works, but later shifted his focus of attention from structural qualities of his work towards the creation of a sound. He built up an ensemble that was crucial in maturing his compositions—the interpretation was given an important role in the making of his music.

Another important aspect of minimal music is the involvement of the listener—as minimalist art focused on the necessity for the viewer to ‚complete the work', the same can be said for performances of minimal music. Brian Ferneyhough noted that „all music is many-layered … Our ears impose on us, with any listening process, a number of possible strategies which we're constantly scanning and assessing and … finding a new distance or new perspective in relation to what we're hearing at that particular moment. It's one of the few possible justifications for minimalist music, for instance: that the maximalisation comes through the individual, rather than through the object" (Brian Ferneyhough, in conversation with David Osmond-Smith during the interval of a recording for BBC Radio Three, first broadcast on 18 July 1993, quoted in ibid., 14f).

The influence of minimalism extends to current pop music, as elements of minimalism were introduced into pop, and even muzak by Brian Eno and Reich himself; a generation of artists that are now creating electronic music have been influenced by the minimalisms. Modern interpretations of minimalism, however, are more easy to listen at, and tend to view the original minimalist compositions as refreshingly extreme, visible e.g. in the enthusiasm of Björk who celebrates minimalism's ability to „shake off that armour of the brain" (Björk quoted in ibid., 46)

2.3 Minimalism Found Elsewhere

As this chapter aims to provide an overview of the broad range of historical definitions of minimalism, it cannot be withheld from the reader that the term ‚minimalism' has not been confined to art and music: it has since found wide application in diverse areas, starting with typography, architecture and literature and reaching as far as food or politics.

However, while some of these uses enrich the definition of minimalism, other applications of the term are motivated by rather superficial aesthetic similarities: using few and fresh ingredients for cooking (Katzen, 2006), or linking the minimal amount of food in French Cuisine to minimalism helps little to further explain the term. Even more serious uses of the term are often only using the concepts transported by minimalism, and not adding to them, e.g. in the minimalist conception of state (Nozick, 1974) that starts with the state-of-nature of the individual and goes little beyond reducing the state to interfere as little as possible (Böhr, 1985, 5 + 121f).

A focus on the original meaning motivated the following, necessarily small, selection of ‚minimalisms': literature and architecture illustrate the meaning of space, and typography adds the conflict between form and function that we will return to when discussing minimal design (compare 4.1).

2.3.1 Literary Minimalism: Roots in Hemingway, Archetype in Carver

> The world is so complicated, tangled, and overloaded that
> to see into it with any clarity you must prune and prune.
> – Italo Calvino, If on a Winter's Night A Traveller

While ‚Minimalism' in the visual arts tried to avoid any implications beyond the object itself, for literature the opposite could be claimed: within a minimal frame, the evocation of larger, unnamed issues is often effectuated by figurative associations. Stella's „what you see is what you see" is contrasted with Hemingway's „tip of the iceberg" aesthetic principle, by which he suggests that seven eighths of the story lie beneath its surface. On the other hand, Minimal art in the post-modern tradition often requires the viewer for completion, and thus shares with Minimal literature an interactive character, the strong involvement of the viewer that John Perreault described as: „The term »minimal« seems to imply that what is minimal in Minimal art is the art. This is far from the case. There is nothing minimal about the »art« (craft, inspiration or aesthetic stimulation) in Minimal art. If anything, in the best works being done, it is maximal. What is minimal about Minimal art … is the means, not the ends" (Perreault, 1967, 30).

As the art in Minimal art is rooted in the very difficulty of extreme reduction, the apparent simplicity of literary minimalism, with its stark prose and tacit narration, established a highly stylized language. Reflection and implication are used as means to express the unstated. This conscious, careful use of language was described by author Amy Hempel: „A lot of times what's not reported in your work is more important than what actually appears on the page". Minimalist short stories in particular often take working class reality (‚blue collar stories') as a subject; at first glance, only everyday activities are described—yet although the text resembles reality, the presentation suggests that there is more to the text than the narrated details. As Cynthia W. Hallett notes, „Minimalist writers … employ an aesthetic of exclusion–a prudent reduction of complex equations, a factoring-out of the extraneous until the complicated is expressed in the simplest of terms. Generally in these texts, distraction and clutter are stripped from human commerce until the reader encounters the whole of society reflected in slivers of individual experience, the unstated present as a cogent force" (Hallett, 2000, 7).

Although scholars disagree about the value of covering different authors „in the same breath" for fear of watering the term (Trussler, 1994), literary minimalism has been traced back to the prose poem developed by Baudelaire and Russian poets such as Ivan Turgenev, Fedor Sologub and Daniil Kharms who created an aesthetically stark language with symbolist, futurist and absurdist miniatures (Wanner, 2003). In American literature, minimalism owes much to the clean and spare literary style of Ernest Hemingway, but Edgar Allen Poe, Anton Chekhov and James Joyce are included as strong influences on the Minimalist style (Hallett, 2000, 12). Edgar Allen Poe's notions of „unity" and „singleness of effect" are both achieved through exclusion: „In the whole composition, there should be no word written, of which the tendency … is not to the one pre-established design … Undue length is … to be avoided" (Poe, 1842). As selective inclusion is achieved primarily through conscious exclusion, secondly through omission of excess, an inner

unity is achieved that delivers the „certain unique or single effect to be wrought out" (ibid., 950). Although he rejected rationality in the tradition of Romanticism, his calculated composition of emotions is a fine example for a rational, artificial process that „combines such events as may best aid … in establishing this preconceived effect" (ibid., 950).

Chekhov continued the exploration of the singular effect using „a plotless design that focuses on a single experience … [and] an objective presentation which so distances the narrative voice that the reader is drawn into closer association with the story" (Hallett, 2000, 31), the latter a typical treat of Minimalist fiction. Joyce—at least in his early works – established a minimal dependence on the traditional notion of plot, his use of seemingly static episodes and ‚slices of reality' is also part of the Minimalist style (ibid., 12). Samuel Beckett and his efforts „to present the ultimate distillation of his inimitable world-view … to compress and edulcorate [purify] traditional genres" (Hutchings, 1986) have also influenced not only Minimalist writers (comp. Glass), yet the major contribution to the Minimalist style could be the „concept of language as an inadequate tool for communication, especially for conveying emotion, subjective concepts and intangible matters", later a „major element of the minimalist writer's sensibility" (Hallett, 2000, 35).

In the late 1960s and early 1970s, a new generation of authors, including Ann Beattie, Frederick Barthelme, Raymond Carver, Amy Hempel, and Mary Robison rediscovered „simplicity and immediacy of fictional expression" (ibid., 10) for themselves and initiated a shift from ‚well-made' literature to a „growing concern with character and sensibility, with the inner dynamics of the landscape of individual psychology … related to the historical shift from moral absolutism to moral relativism as the dominant sensibility in our culture" (Weaver quoted in ibid., 126). The forms that minimalism takes in literature are as diverse as its authors: Barth notes that in Literature, minimalism can be of „unit, form and scale", of „style … vocabulary", or of „material" (Barth, 1986).

The final meaning of a Minimalist story unfolds only within the personal context of the reader—as ‚simple' facts are interpreted, „metonymic matter is transformed into metaphoric signifiers" (Hallett, 2000, 11). Strong psychological, social and historical associations are generated by the indirection and understatement of the deceptively simple figurative language of minimalist prose. What appears to be a single event, or a depiction of ‚nothing happening' signifies human condition or capacity—a connection between the trivial and the significant is created. Also, what is omitted becomes prominent; often, the inner state of personas is communicated without speaking the exact words, but rather relying upon „ready phrases, expected responses … or euphemism to say it any other way but outright" (ibid., 19). Just as one could claim that nothing exists unless it has a name, by alluding the exact definition of inner feelings and motivations, they ‚keep their innocence', or with Sartre: „To speak is to act, and everything we name loses its innocence, becoming part of the world we live in" (Sartre, 1965).

Minimalist writers do not try to achieve closure within the narrative, but rather compose story elements for final assembly by the reader. Thus, the sentences, the ending as much as the first lines, often seem disconnected, the works are „but shells of story, fragile con-

tainers of compressed meaning" (Hallett, 2000, 11). This results in an unusual open-endedness of most of the short stories and in a rejection of linear plots; even more, this illusion of a ‚storyless' story often goes along with a seemingly ‚authorless' story, as the reader seems to overhear a conversation, or eavesdrop on an event, rather than being told a story: the „suppression of the artist's personality can be virtually total … [the abnegation of individual style is so complete that … we cannot tell one writer's work from another's; yet the very suppression of style is a style–an aesthetic choice, an expression of emotion" (Gardner, 1991, 36).

The label minimalism also implied some negative meaning in literature, some critics drew a curtain between prose poetry as an art and short fiction as a mere craft that did not allow them to appreciate the carefully crafted language of Minimal fiction. Jerome Klinkowitz was convinced that minimalist writers did not dare to judge human behavior and lost the ‚artfulness' of storytelling in their oversimplified descriptions, that Minimalist fiction „suspends all aesthetic innovation in favour of parsing out the parsing out the most mundane concerns of superficial life (for fear of intruding with a humanly judgmental use of imagination)" (Klinkowitz, 1993, 364). Sven Birkerts was dissatisfied with the lack of resolution and asked for fiction „to venture something greater than a passive reflection of fragmentation and unease" (Birkerts, 1986, 33)

Barthelme gave the following advice to those ‚convicted' of minimalism, again highlighting the importance of the reader, and the virtue of being open for interpretation: „Tell them that you prefer to think you're leaving room for the readers, at least for the ones who like to use their imaginations; that you hope those readers hear the whispers, catch the feints and shadows, gather the traces, sense the pressures, and that meanwhile the prose tricks them into the drama, and the drama breaks their hearts" (Barthelme, 1988, 27).

2.3.2 Minimalism in Architecture

Minimalism has also been used to describe the architecture of human dwellings. In this discipline, a fairly clear line can be drawn between those who favor a ‚minimalist look' because of its clear lines and impressive shapes, and those who focus on a minimal visibility of architecture for its users. As perhaps the most prominent representative of the former approach, Ludwig Mies van der Rohe adopted the motto „less is more"; for him, this meant flattening and emphasizing the building's frame, eliminating interior walls and adopting an open plan (cf. Schulze, 1986). Thus, he aimed to reduce the structure to a strong, transparent, elegant skin—he termed this „skin and bones" architecture.

The clear forms of minimalism in architecture were interpreted as a reaction to the attention-calling visual excess of the supermarket culture. Minimalism is still a label that is broadly used for current industrial and civil architecture. The use of the label itself is seldomly discussed, and in contrast to other disciplines, definitions are given using a multitude of examples rather than a formal explanation (Cuito, 2002b; Cuito, 2002a; Petterson, 2003; Petterson, 2004; Castillo, 2004). Minimalism is also used as a term to describe

the outer surroundings of buildings, gardens (Levy, 1996; Bradley-Hole, 1999) and public spaces (Cuito, 2001), and interior design (O'Bryan, 2002; Rossell, 2005).

Figure 27: Claudio Silvestrin: Neuendorf House, Mallorca (1989).

Architecture tends to shy a standardization almost as much as art. Combining the clear lines of minimalism with great visual variation, maximalism evolved in turn as a reaction to minimalism in architecture (compare Figure 28). This counter-reaction was based on a rejection of the total reduction of forms and colors that threatened to create a sterile environment. In *Learning from Las Vegas*, Venturi (1972) argued that within the visual chaos of Las Vegas, signs are not any longer only decoration, but acquire a central function in architecture. He criticizes minimalism succinctly: „Less is not more. Less is a bore." (ibid.).

Figure 28: Maximalism: Peckham Library (1999), by Will Alsop.

Although these new decorative elements have strongly influenced modern architecture, minimal architecture has not gone out of fashion. Yet, as architects Pawson and de Moura argue, it has gone beyond the simple geometrical forms, and moves „towards concrete attempts, albeit thinly scattered over time and space and in modest quantities, to introduce a life more imbued with spirituality, clarity and harmony" (quoted in Bertoni, 2002, 149ff). This tendency towards integration of existing spaces is often marked by the integration of existing architecture.

As architect John Pawson notes, in minimal architecture, „Emptiness allows us to see space as it is, to see architecture as it is, preventing it from being corrupted, or hidden, by the incidental debris of paraphernalia of every day life" (quoted in ibid., 134ff). Architect Michael Gabellini adds: „Many people think that Minimalist art or architecture is something cold, abstract and sterile. Instead, minimalism is not only art or architecture, actually is is an idea that does not elude existence. It is analoguous to the editing of a film, where there is an inherent concentration of form and experience. More than a subtraction, Minimalism is an inherent concentration of experience and pleasure" (quoted in ibid., 183ff). Franco Bertoni highlights the contribution of minimalism to the discipline, namely a heightened awareness for space: „Leaving aside present-day misuse and the inflation of the term, Minimalist architecture represents one of the most significant contributions to a review of a discipline, and an attempt to endow it with new foundations, and a way of life" (ibid., front flap).

As in other disciplines, minimalism connotes a collection of works of architects from profoundly different origins and cultural backgrounds; common to most approaches is primarily the rediscovery of the value of empty space, and extreme simplicity. Bertoni further lists „a reduction in expressive media, and a radical elimination of everything that does not coincide with a programme" as characteristics for minimalist architecture, often resulting in „minimalistic design overtones, … and formal cleanliness" (ibid., front flap).

2.3.3 Minimalism in Typography

Although the use of the notion of minimalism is not agreed upon within typography, and terms such as modernist and functionalist typography are often used interchangeably, again an immediate and a more reflective understanding of minimalism can be observed. The direct translation of ‚minimal' is exemplified by designer Ron Reason's claim that minimalism results in better typography: „minimalism in typography translates immediately into cleanliness, orderliness, easy of navigation, and consistency" (Reason, 2001). Here, minimalism is understood as the reduction of means–the use of less typefaces.

In contrast, the more reflective understanding results in the use of less, and less ornamental typefaces and the accented use of whitespace to structure content; its foundations are rooted in the conceptual shift of typography as functional design—following the development in the liberal arts. According to Ferebee (1994, 107), „two extensions of Cubism—de Stijl and Constructivism—were the primary influence on Functionalist typography and poster design". The work of de Stijl and Bauhaus typographers was consolidated in Tschichold's (1928) *The New Typography* (1994, 110).

Tschichold summarizes the intention of his typography in three sentences: 1. The new typography is purposeful. 2.The purpose of all typography is communication. 3. Communication must be made in the shortest, simplest, most definite way“ (quoted in Good & Good, 1995). This striving for functionality has been understood as a reaction to on the one hand the possibilities opened by mass production–much as the minimalist artists auseinandersetzten with artificiality and new materials–and on the other hand the increased scale of the consumer market for printed articles. Seeking solutions to communication problems, typographers around Tschichold tried to enhance effective communication. This interpretation is exemplified by Katherine McKoy, historian of graphic design: "Modernism, especially at the Bauhaus, was a response to the economies of scale and standardization in the new mass societies. This functionalist design philosophy of 'form follows function' is based on standardized processes, modular systems, industrial materials and a machine aesthetic of minimalist form. Universal design solutions were sought to solve universal needs across cultures" (McCoy, 1995).

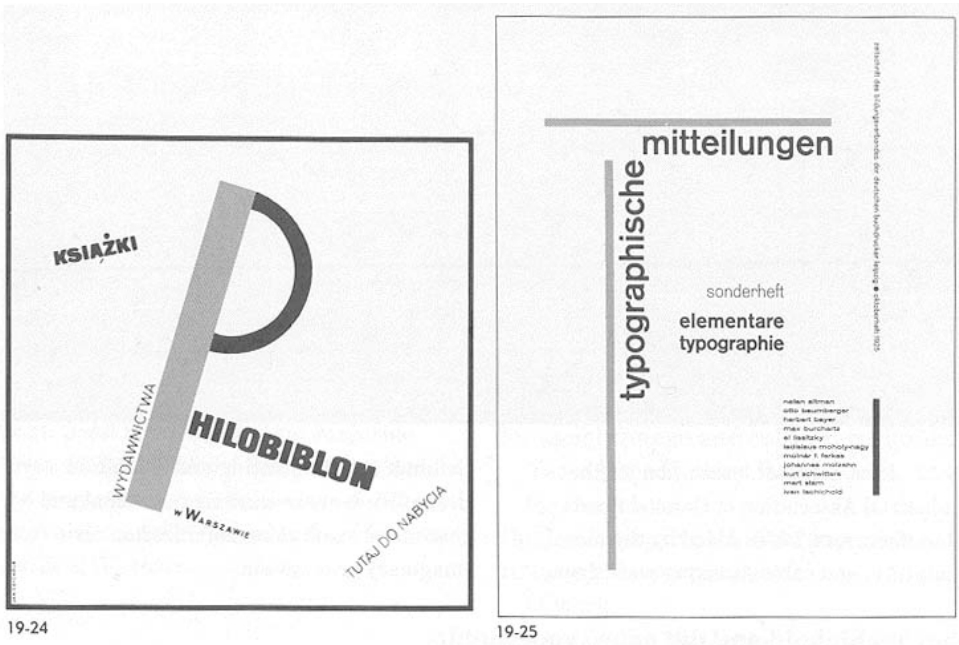


Figure 29: Typographic design by Jan Tschichold (1925).

Tschichold understood himself as a typographic designer, a profession that did not exist before, and that aimed to create graphic designs using typefaces. Previously, typesetters followed the rules of their trade, and understood typesetting as a craft that tried to create products that excelled merely in their usability. *The New Typography* has been understood as technical-symbolical formalism, providing rules and examples for creating typographic designs. It is not. As Tschichold notes, „Merely to copy its external shapes would be to create a new formalism as bad as the old. … It will come only from a complete reorientation of the role of typography and a realization of its spiritual relationship with other activities“. (Tschichold, 1928, 7). And further, „Despite its many illustrations, this is not a copybook. It is intended to stimulate the printer and make him aware of himself and the

true nature of his work" (ibid.). Tschichold emphasized the conceptual frame that is the basis for his designs; his minimalism is of more than superficial nature.

A number of influential minimal designs were created by Tschichold in the 1920s and 1930s, and the minimalist font *Grotesk* became popular, yet Tschichold himself returned to the use of ornamentals in his later work period, and also started to using serif fonts, creating the Sabon face, a Garamond variant, in 1964. After having to emigrate to Switzerland, he slowly acknowledged the *function of ornamentals* in typography and the benefits of creating a pleasant and less harsh layout (Aynsley, 2001, 68). One could also argue that he never became ‚playful' (in a negative sense) and always kept true to his sense of functionalism as „serving a communication purpose" (Tschichold, 1928). His work with Penguin Books, for whom he overhauled the typography of their paperback series, is visible to the day.

2.4 Homing in on Minimalism: Summarizing the Art perspective

If the doors of perception were cleansed
everything would appear to man as it is, — infinite.
— William Blake, The Marriage of Heaven and Hell

Although this chapter started out contesting the feasibility of a universal definition for minimalism, it attempts to close with a short synopsis of the characteristics that have been subsumed under this label. Although almost every use of the meaning of the term minimalism refers to a unique meaning, some common motives can be identified that appear in several contexts. Among these repeating motives in minimalism are a *minimality of means*, and a *minimality of meaning*, a *minimality of structure*, the repeated *use of patterns* and the focus on *involvement of the recipient*. The following chapter refers to these motives as it redefines minimalism within the context of designing of interactive systems.

Before this chapter concludes with a description of the identified motives, however, a short note on methodology must be allowed. Computer science has a tradition of unscrupulously borrowing from other sciences. In its beginnings, this need arose from the novelty of the subject, and many methods that seemed to prove useful but originated in other disciplines were quickly adopted to form the new core of computer science. This thesis does not try to integrate or assimilate yet another discipline, instead a more modest aim is followed: minimalism as a concept from the liberal arts is borrowed to illustrate existing practice within computer science. Although computer science tends to prefer simplifications if they generate working models suitable for implementation, this terminological transfer cannot yield engineering rules, but rather promises to deliver a critical perspective that helps to find and judge design decisions.

Even if a definition of minimalism existed, the transfer into computer science would require both interpretation and speculation. However, within the liberal arts there is a rich literature describes an even richer microcosm of definitions for minimalism, and subtle differences are considered valuable. The notion of minimalism in art is descriptive—it cannot be used to create art. This is partly owing to the definition of an artist—artworks

cannot be constructed using blueprints (and where they can, e.g. in concept art, the blueprints assume the role of the artwork), and artists don't follow rules, they follow their inner musings. And it is partly caused by the wealth of different styles that are subsumed under the term minimalism.

2.4.1 Minimality of Means

A recurring issue in minimal art and music is the choice of *minimal means*: painting with enamel paint, choosing monochromes as a medium, tap-playing a desk, or a single-string instrument. Whether this minimalism can be interpreted as a reaction to pop art's excessive use of symbols in the monochromatic paintings of Klein, Reinhardt, Rauschenberg and others or the static harmony, steady beat and static and sparse instrumentation in Glass's and Young's early compositions, minimal materials are consciously chosen as the basis for artworks. Minimalist plastic art was constructed with the most simple shapes, cubes or cuboids. This motive is perhaps the least controversial, and often Young is quoted to have defined minimalism as „that which is made with a *minimum of means*." (quoted in Schwartz, 1996, 7)

2.4.2 Minimality of Meaning

Before minimalism, painting was dominated by Abstract Expressionism that is related to Surrealism in its focus on emotions. Minimalism was considered much more impersonal, and took abstractness to the next level—some paintings had neither title nor intended meaning, and great care was taken to avoid visible traces of the artist and his intentions. „What you see is what you see" summarizes this approach to both painting and sculpture—there is no more to art than can be seen. The usual „function" of art, the expression or embodiment of values, visions, or emotions, was reduced to the maximum. In music, the interpretation of *minimality of meaning* was different; music was used to create hypnotic stances, or a singularity of mood, e.g. by Young in his drone period. Here, the „function" of music was to create a singular feeling, or, expressed negatively, monotony—often with the intention of sharpening the listener's awareness. In other fields, a similarly strong focusing effect can be observed, e.g. in architecture or literature, where open spaces and ellipses work to concentrate the attention of the observer.

2.4.3 Minimality of Structure

Not only the materials of minimalism are often minimal, it is also their combination in the artwork that is minimal in *structure*. Both in art and in music, the exposition of the inner structure is purposefully executed, whether in Young's additive rhythms or Judd's square-based sculptures. Simple rules and repetition are used to make the structure transparent for the recipient.

A different approach is taken in minimal literature: here, structure is limited by the formal choice of the short story as a medium, minimizing the textural complexity found in other literary works—without ado, the reader is plunged deep in the story. The structure of minimalist literature is often thoughtfully composed, yet readers are encouraged to

simply follow the author's lead without reflecting about a text's structure. Instead, they are immediately confronted by topic and message. Similarly, minimal typography aims to reduce decoration and distraction, thus reducing cues for the visible structure of the layout. And even in painting, structure is not only minimized to make its underlying principles transparent to the recipient. Often, the minimization of structure follows the objective of finding a balance of inner structure and the artwork's other qualities. Judd described this as „Take a simple form—say a box—and it does have an order, but it's not so ordered that that's the dominant quality. The more parts a thing has, the more important order becomes, and finally order becomes more important than anything else." (Glaser, 1966, 156)

2.4.4 Use of Patterns

Repeating elements have been found in both art and music long before minimalism. The excessive use of patterns, however, introduced a new quality and differentiates minimal music from other „contemporary classical" music. This use of patterns is closely linked to the question of control: On the one hand, control is lost as not pieces but only patterns are composed and combined with rules for correct combination or progress. On the other hand, some minimalist composers expose a tendency to completely control not only the performance but also the form of reception. Between these extremes, composers give performers a before-unknown freedom of interpretation within a strict rule system. Through additive processes and phase shifting, the patterns—even if simple in themselves—can be used to create very complex sound textures. In minimal art, serial paintings use small variations to create an overall impression; again, the combination of individual pieces is more complex than the individual works. When recipients learn to identify these patterns, they learn a new perspective that focuses on the combinatory possibilities hidden in the work of art.

2.4.5 Involvement of the recipient

„Serious art" is usually „displayed" in art museums, special places built specifically for the enjoyment of art. The division of roles is clear: a piece of art is shaped and given a meaning by the artist, the consumer, or „art connoisseur", can observe and make his interpretation, but he is clearly separated from the artwork. By stripping painting from everything that is usually connected with meaning, by presenting pure white surfaces, minimalist artists forced the artwork to take on meaning through observation. Minimalist art is different from concept art, where a blank sheet would not reflect the onlooker's thoughts but would be based on the artist's (possibly most complex) conception. Minimal art cannot stand for itself, it is not depictive, yet it also does not require explanation. This consciously unartistic method of producing art creates a new freedom of meaning for the spectator, who in turn becomes part of the artwork.

2.4.6 The Minimalist Perspective and Criticism

Many artists that were associated with the label of minimalism did not agree. Partly, this is because artists never like labels, they tend to understand their work as unique, and consequently resist categorization. And partly, it is because minimalism is understood as a negative term. It denotes the minimal—that which is so reduced that there is nothing that could be less.

Minimalism is thus often linked with the obscure and simplistic. This critique, however, does not extend deep enough. Minimal art seems simplistic only if the effect is examined without observing the cause—minimalism is as much a set of mind as it is a set of practices. A canvas painted with pure white enamel paint, without traces of the artist remaining, is meaningless without the conception of the artists behind it. For the artists as for this thesis, this theoretical conception behind minimal works is crucial.

However, this highlights an important quality of the minimalist perspective: it draws attention to the extreme. A key ‚feature' of the notion of minimalism is its ability to draw critique. As modern electronic musicians, designers must decide how far towards the minimal they dare go.

3 A Role for Minimalism in the Use-Centered Design of Interactive Systems

This chapter lies at the heart of this thesis: it puts forward four notions of minimalism as tools for the analysis and design of interactive systems. The notions of minimalism presented here can be used to describe *motivations* underlying use-centered design; they focus on a particular set of goals for human-computer interaction shared by experts who would otherwise hesitate to agree with each other. While this role could also be fulfilled by the related notion of ‚simplicity'—an often-used value in human-computer interaction, the latter is, as an exclusively positive term, difficult to criticize, and thus less useful for analysis and reflection. The four notions of minimalism presented here promise a more accurate understanding of the what and where of reduction and suggest themselves for a more detailed analysis of design.

Before introducing the perspective that is created through the transfer of minimalism to design, human-computer interaction (HCI) is defined as the target domain of the transfer (3.1), and differences of the new terminology and both simplicity (3.2.1) and existing concepts of minimalism (3.2.2–3.2.5) are examined to minimize the risk of misunderstandings. Consequently, functional, structural, architectural and compositional minimalism are defined as four notions of minimalism (3.3.1–3.3.4), and their interrelationships are discussed to form a minimalist terminology.

3.1 Meanings of Minimalism in HCI — a Transfer from the Arts

Although the design of interactive systems is sometimes called an „art" (Laurel, 1990; Bickford, 1997; Cameron, & Limited, 2004), this is done more in mockery than in admiration. Design is often seen as a craft (Wroblewski, 1991)—and for commercial applications, designers of interactive systems often have a training that is unlike the instruction of artists, or even designers: a team of „interface designers" can consist exclusively of technically trained (software) engineers. Engineering typically seeks to provide optimal solutions for well-specified problems: the Encyclopedia Britannica defines engineering as the

„professional art of applying science to the optimum conversion of the resources of nature to the uses of humankind" (Britannica, 2003). The specification of requirements for complex interactive systems represents a new dimension of complexity (Pohl, 1997; Nuseibeh, & Easterbrook, 2000). As it is difficult to perfect systems for a constantly evolving and inherently imperfect context, this has lead to approaches that separate the computational from the contextual: Dijkstra (1989; 1981) used the metaphor of a firewall that would separate all imperfection from the programmer's view. However, this only relocates the responsibility of dealing with real-world applications, and in practice, ignorance of the context is not helpful (Coy, 1997)—a „design" approach is required that considers the activities and needs of end users (comp. Oberquelle, 2005; Winograd, & Flores, 1987). Floyd (1993) introduced a process perspective to software engineering that is based on the assumption that design is not limited to the product, but that the process of design must also be object to active development during software construction (comp. Floyd, 1994).

This balance between the technical engineering side and the artistic designer's view is a difficult one (comp. Mackay, & Fayard, 1997; Wolf, Rode, Sussman, & Kellogg, 2006), and requires this text to follow both a more formal approach, and the abstract perspective on design. To satisfy the former, the object of study must first be defined more clearly. This starts with the context in which the notion of minimalism is to be used: the design of interactive systems. While the definition for interactive systems is rather undisputed (definition 3.1), the engineer's perspective manifests itself already in the terms „user" and „user interface" (definition 3.2).

> ***Interactive systems*** are a class of information processing systems where control is exerted *by users* in an *interactive* manner.

Definition 3.1: Interactive systems

> The interaction between a user and an interactive system is mediated by a ***user interface***, which comprises *all parts*—both hardware and software—of the interactive system *with which the user is able to interact*. (comp. Moran, 1981) The term user interface is elsewhere sometimes used in a broader sense that includes the aesthetic appearance of the device and the content that is presented to the user within the context of the user interface (e.g. Preece, 1994).

Definition 3.2: User Interface

> ***Human-Computer Interaction (HCI)*** is the process of *information exchange between a user and an interactive system*. The quality of HCI is usually measured based on the effectiveness and efficiency in accomplishing predefined goals for (work) tasks and the satisfaction of the user. (ISO9241, 1998)

Definition 3.3: Human-Computer Interaction

Grudin raises the point of distinguishing between „»the user interface« to a computer and »the computer interface« to a user or users" (Grudin, 1990a, 1): the normal, „technology-centered" use of the term, which is also adopted in this thesis—the *user* interface is seen as part of the *system*—shows that the engineer's focus on the system is prone to mask requirements of the user (Grudin, 1990b); the problem is even more severe for groups of users and their requirements (Grudin, 1990a).

A more complete view on user interfaces has been advocated by numerous disciplines, making HCI (definition 3.3) an intensively interdisciplinary field (comp. Myers et al., 1996; Rozanski, & Haake, 2003): to provide some examples, psychologists have increased awareness of „human factors", cognitive scientists stressed the relevance of information processing and learned to value (Winograd, & Flores, 1987), perusing anthropology, ethnomethodological observation techniques were introduced to study the behavior of users in their natural habitats (Suchman, Pea, Brown, & Heath, 1987; Button, & Dourish, 1996; Martin & Sommerville, 2004), following activity theory lead Bødker to stress that „users interface through computers with their work" (Bødker, 1990), and distributed cognition theory was used to explain the role of environments (Hutchins, 1996; Hollan, Hutchins, & Kirsh, 2000).

In this study, the term *use-centered design* (comp. Vredenburg, Mao, Smith, & Carey, 2002) will be used as it integrates the two before-mentioned notions of interface, and the examination of user needs as well as the analysis of technological possibilities in a single term (definition 3.3). Use-centered design is also not only interested in the user and her well-being, it assumes a holistic socio-technical perspective that includes other tools, and work practices to contextualize the user's needs, her perception, intentions and goals. Minimalism is here used as a term to analyze aspects that touch both the users' goals, needs and tasks and the material structure and technological architecture of design artifacts—following the tradition of industrial design, the user interface and the machine interface, the task structure and the system structure are taken to be interdependent.

> ***Use-centered design*** focuses on the goals and tasks associated with the use of technology. As user-centered design, it tries to structure the functioning of a user interface around how people can, want or need to work, rather than the opposite way around. The focus is, however, not on the user alone, but also on the tool and the task. (cf. Flach & Dominguez, 1995)

Definition 3.4: Use-centered design

3.2 Defining the Scope of Minimalist Terminology

The previous chapter might have provided some hints the applicability of the term minimalism to different disciplines. A term as broad as minimalism evokes different interpretations, and while all of the following are legitimate, some are simplifications only useful in special contexts, while others are both different from the notion of minimalism quoted so far and difficult to apply to design. The sketches of different meanings of minimalism that now precede the formal definition of the term are differentiations from what this

thesis does *not* understand as minimalism—placing these meanings in relation and explicating the transfer from liberal arts to interaction design.

3.2.1 Illustrating the Limits of the Notion of Simplicity

Simplicity is often an aspired value in the design of human-computer interaction. The positive nature of simplicity is obvious: „simplifying" the user experience—removing hassles and unnecessary tasks, making the task at hand and life generally easier—will improve usability. However, to make simplicity more than a symbolic value, it would have to satisfy the same criteria that are here put forward towards minimalism—it must make a useful tool for analysis and possibly for construction of interactive software.

This, however, remains unproven with as undefined a term as simplicity: often, circular reasoning suggests something is usable if it is simple, and it is simple if it is usable—or a definition is omitted completely as in Nielsen's *The Practice of Simplicity* (2000). As a purely positive measure, simplicity is difficult to criticize; this prevents the discussion of the negative traits of simplicity as, clearly, others might find that the sophisticated is preferable to the simple. Questions of direction and degree, „what shall be simplified?", and „when should one choose the complex over the simple?", cannot be easily answered—and simplicity is also suspiciously subjective in its focus. Furthermore, interactive systems are complex in themselves. Considering their context further increases complexity, and simplicity in design can only be procured for certain, well-defined aspects. Simply stated, while simplicity can be a useful value to pursue, it is not a helpful guide in pursuit. Minimalism, however, is—rather than a value—a means of describing different ways of reaching simplicity.

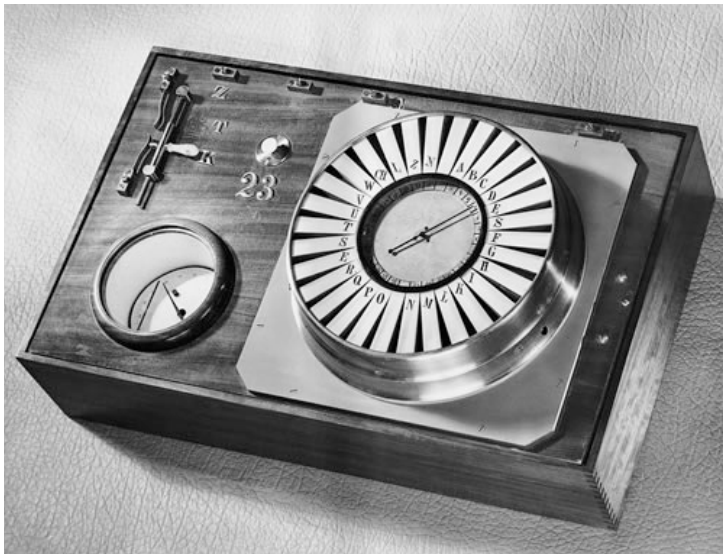


Figure 30: ‚Zeigertelegraph' designed 1846 by Werner von Siemens, built by Johann Georg Halske.

As an example, consider a fundamental task of today's information society: instantaneous information transmission over large distances. When first implemented in the form of electric telegraphs, different technologies were competing that built on the electromagnet

demonstrated by William Sturgeon in 1825. The european version of the telegraph was patented 1837 by William Cooke and Charles Wheatstone in Great Britain. If featured what is arguably the most simple interface to transmit english letters over a wire: a dial to choose a character, and a button to transmit a sign indicating that the chosen character is to be received and added to the transmitted text. Although the Wheatstone's first telegraphs needed up to five wires for transmission, an improved version was designed and consequently marketed by Werner von Siemens in 1847, needing only a single transmission wire (Figure 30). Unknown to the user, small electric impulses were transmitted as the dial was turned, thus enabling the receiver's dial to copy the motion of the sender's dial.

This „simple" interface was competing during the second half of the 19th century with an interface that was simple in different ways: the morse telegraph. The morse telegraph uses but a single key to encode characters in sequences of dots and dashes—or short and long electric impulses sent over the wire. A predecessor of the groundbreaking morse code was demonstrated in 1838 by Alfred Vail (Pope, 1888; Calvert, 2004), but it was his partner Samuel Morse who became famous as he built the device used in the first continental demonstration of delivering a telegraph message from Baltimore to Washington in 1844 (Figure 31). While the efficient use of morse telegraphs is a learned skill, the morse alphabet was designed so effectively to match the task of transmitting english texts that not only the reliability of transmission but also the speed of transmission exceeded those of the dial telegraphs (the speed of professional morse telegraphs exceeds 20 words per minute).

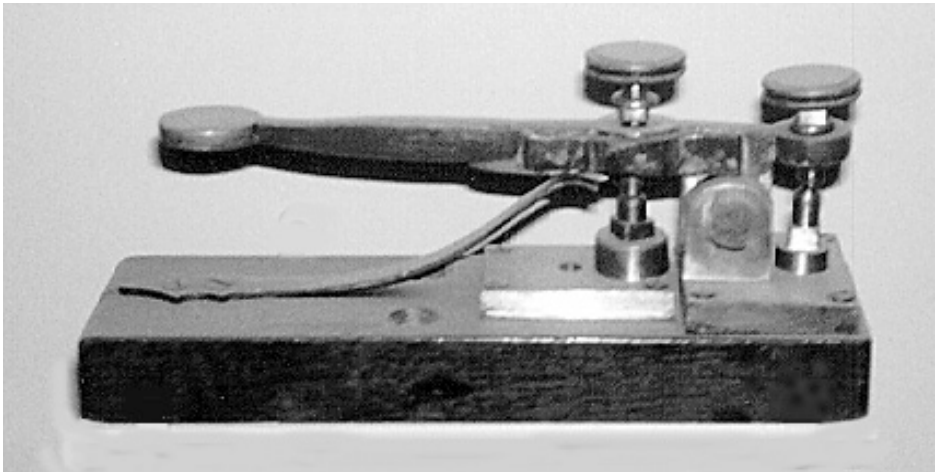


Figure 31: The original „lever correspondent" used in the 1844 Baltimore-Washington demo.

The dial telegraph offered a direct mapping of interface and functionality and was easy to use. Yet is was slow and in contrast to the morse telegraph it could not transmit other characters than those conceived of during construction. The morse telegraph is constructed in the most simple way imaginable, its mode of operation is immediately transparent, it is simple to extend the alphabet, and easy to adapt to other uses (even music Tulga, & Tulga, 2005). Which of these interfaces is simpler?

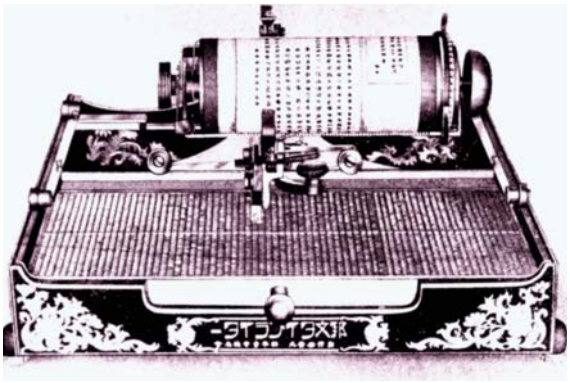


Figure 32: First Japanese typewriter designed by Kyota Sugimoto (1929).

To clarify this point, Japanese text input serves as an illustrative example: equivalent to the dial telegraph are approaches that try to build a Japanese‚keyboard' (see Figure 32). In order to construct typewriters able to print *kanji*, the smaller, yet still numerous subset of chinese characters used in japanese language, 2,400 characters were chosen as a result of analyzing their frequency of use in public documents. The characters were arranged by classification on a character carriage, and the chosen character was raised by a type bar and typed against a cylindrical paper supporter. This typewriter was patented by Kyota Sugimoto in 1929 and consequently used for the production of office documents. As a consequence, typists rushing out to the next typewriter store, trying to purchase new letters not included in the standard set (often used in names) were a common sight in the 1930s (Jun'ichiro, 2001). However, typing was still slow, skilled typists could manage to type up to 2 characters per second, about twice the speed of handwriting.

Figure 33: Chinese keyboard prototype, National Chiao Tung University, Taiwan.

Efforts to produce a keyboard for chinese characters did also fail to gain wide adoption—direct access to all letters becomes more difficult as the number of letters increases (Figure 27), and chinese newspapers require knowledge of around 4,000 characters while dictionaries can contain up to 50,000 characters (Zhou, 2003, see Figure 33).

Japanese and chinese text input nowadays is a compromise between the number of buttons and the complexity of combining different keystrokes. The keyboards used for Ja-

panese and Traditional Chinese are based on the standard 106 key keyboard, adding two letter keys (from within the backspace and right shift keys) and function keys along the space bar. These extra function keys are used to control the character composition (e.g. with the Microsoft Input Method Editor, Rolfe, 2003, Figure 34).

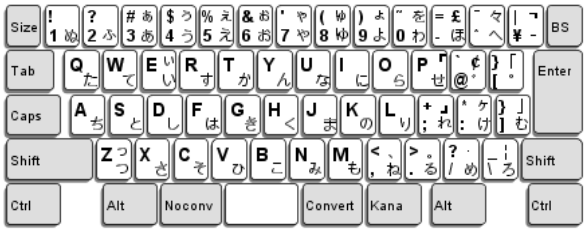


Figure 34: Kanji Keyboard Layout for Microsoft Windows Operating Systems.

John Maeda of MIT, an advocate for simplicity in HCI design, summarizes the problem of defining what is simple with a clever redefinition of simplicity: „Simplicity is not about living at the poles of simple versus complex. It is not about being zero or none. Instead, it is about some." However, with this redefinition in his „tenth law of simplicity", Maeda (2005a) makes finding flawed uses of simplicity impossible. Furthermore, simplicity is used as a value that is ascribed to objects after it has been proven that they work; the term thus has neither explanatory function nor does it spend advice on design extending beyond the examples[2].

3.2.2 A Differentiation from Mathematic Minimalism

Simplicity is prerequisite for reliability.
—Edsger W. Dijkstra (Dijkstra, 1982)

In mathematics, and also in theoretical computer science, minimalization of terms is an important and often used method to create *readable* and *unambiguous* formulae (Cornell, 1997). Reducing the visual and functional complexity of terms, and thus increasing the readability of mathematical calculations, is central to manual mathematics. Reduction is used to unambiguously identify equivalent forms; for example in the search for prime numbers, or when computing the minimal polynomial of algebraic numbers—identical for complex conjugates—eases the proof that numbers form a ring.

In more general terms, minimalism here denotes a principle that increases human understanding of formal notations: reduction is used to create simplicity and distinctness. This principle is long in use—it has become known as *Ockham's razor*[3], the medieval rule of economy, that „plurality should not be assumed without necessity" (Britannica, 2003). This *Law of parsimony*, as it is also called, suggests the simplest of two or more competing theories is preferable and that an explanation for unknown phenomena should first be at-

2. Instead, as a designer, Maeda uses Gestalt laws to explain phenomena found e.g. in the iPod design.

3. This rule is often cited in latin as ‚entia non sunt multiplicanda praeter necessitatem', and also found under the name ‚Occam's razor'.

tempted in terms of what is already known. Ockham's razor can be interpreted to relate closely to the functionalist understanding of minimalism—a single way of doing is sufficient. Its application to engineering, where it is often invoked to thwart unnecessary complexity, has become famous under the acronym KISS ('keep it simple, stupid').

A different aspect of mathematical minimalism is the „elegance" of mathematics, suggesting an „inner beauty". Hardy famously argued in *A Mathematician's Apology* (1940, 63f & 85) that aesthetic considerations are sufficient to justify the study of pure mathematics– the same argument is ascribed to Poincare: „The mathematician does not study pure mathematics because it is useful; he studies it because he delights in it and he delights in it because it is beautiful." (*Science et Méthode* 1908, quoted in Huntley, 1970) An aesthetic experience of mathematics is shared by many experts; Dirac wrote that it is more important to have beauty in one's equations than to have them fit the experiment, Hungarian mathematician Paul Erdős expressed his views on the indescribable beauty of mathematics when he said „Why are numbers beautiful? It's like asking why is Beethoven's Ninth Symphony beautiful." (cf. Davis, Hersh, Marchisotto, & Rota, 1995, 168-171) Aesthetics are also important for the inner workings of the discipline–proofs are being accepted partly based on aesthetic criteria[4]; however, the aesthetic dimension of a proof can sometimes be retraced to a better match with the essence of the problem: a proof is beautiful when it „reveal[s] the heart of the matter." (Davis, Hersh, Marchisotto, & Rota, 1995, 299)

3.2.3 A Differentiation from Linguistic Minimalism

Noam Chomsky has been heralding several shifts of linguistic research in the last 35 years. While „autonomous syntax" (e.g. Chomsky, 1968) gained immense popularity in the 1970s, the latest turn of his investigation of the human languages and their specific characteristics was given a new name and a more stable form with the publication of *The Minimalist Program* (1995). Its basic assumption is the existence of a single universal grammatical system for all languages, capable of generating different languages by following different choices made from pre-defined options within the system. The minimalist program is then defined as the perspective how propositional thought is linked up with sound, the resulting research questions encompass (1) the function of language given its cognitive and biological environment, (2) the construction of a system that fulfills the core functionality of language–the sort of machinery „a superbly competent engineer might have constructed, given certain design specifications" (Chomsky, 1997, 15) and (3) the comparison of actual human language with this ideal language system.

Chomsky's minimalist approach, dominating linguistic research for decades, was aimed at the construction of a perfect, minimal system. Thus, linguistic minimalism can be interpreted as an attempt to find regularities behind the rules that seem to govern language generation, introducing a more systematic and analytic approach. This ambitious enter-

4. For a sample of beautiful objects in mathematics, see (Huntley, 1970), (King, 1992), (Lang, 1985), (Rothstein, 1995) and (Chageux, & Connes, 1995).

prise has however also been contested, and was criticized as being largely unsuccessful, lacking empirical support and theoretical coherence (Seuren, 2004). Chomsky himself recently reverted to speaking of language as „expressing thoughts" (Chomsky, 1997, 1; Chomsky, Belletti, & Rizzi, 2002, 45), a notion „alien to the random-generator concept of grammar" (Seuren, 2004, 4) and consistent with the rivaling approach of generative semantics. Also, determining language as a „perfect system" (Chomsky, 1995, 1) is complicated by the evolutionary nature of the external systems that language interacts with, resulting in changing conditions for perfection; furthermore, evidence exists that suggests language is often used in „distinctly anti-functional" ways (Seuren, 2004, 19).

3.2.4 A Differentiation from Documentation Minimalism

Avoid missing ball for high score.
– instructions for PONG[5]

Of all reported uses of the term minimalism, John Carroll's minimalist approach to technical documentation, defined in *The Nurnberg Funnel* (Carroll, 1990) and refined in *Minimalism beyond the Nurnberg Funnel* (Carroll, 1998), is closest to the field of usability. Advocating the reduction of learning material, Carroll chose a learning style where „knowledge" is not presented in a pre-structured format, and where users need to actively work on real tasks to acquire knowledge. The underlying proposition is that learning yields better results if the knowledge is actively acquired while being engaged in the context of real work. According to Carroll (1990, 77f), „the key idea in the minimalist approach is to present the smallest possible obstacle to learners' efforts, to accommodate, even to exploit, the learning strategies that cause problems for learners using systematic instructional materials".

Minimalism can thus be interpreted as a use-centric approach to learning: is tries to support the user while she tries to accomplish a genuine task. As support is only necessary where the user fails to achieve a set goal with the available tools, minimalism consequentially tries to present help when breakdowns become apparent. Instead of providing structured material that is designed optimally to convey the system designer's view of the application, minimalism tries to meet with the user in his work. This requires an understanding of his previous knowledge and an adaptation of the knowledge to be conveyed to the user's viewpoint. Carroll (1998) notes that he conceived his notion of minimalism after reading Dewey, Piaget, and Bruner.

While minimalism is aptly named—Carroll undid the focus on structure inherent in classical manuals and replaced it with an on-demand approach, or, with reference to minimal art, an „objectness"—the declared focus of Carroll's minimalism is on computer documentation and learning. As the realms of software development and technical writing

5. PONG is the name of what is commonly referred to as the „first video game" and was produced by Atari (Winter, 1999). The mentioned inscription was the last of three lines serving as „the first minimal manual", the other two being „Deposit Quarter. Ball will serve automatically".

only seldom communicate, his minimalism touches only insofar upon system design as documentation is considered a part of the design. Also, Carroll's notion of minimalism is defined in relation to the comprehensiveness of technical manuals: the new, minimal manuals promise to take up only a fraction of the space of classical menus; thus, minimalism is not an extreme, but an ideal.

One can, however, try to draw connections from Carroll's later work on design methodologies, and specifically on scenarios, to the foundations he laid out in minimalism (6.2.1). Carroll himself remarked in an interview with Dubinsky (1999): „It is funny to talk about the early minimalist work because it really is historical; it is like talking about the Renaissance. I mean it is not going to come again; we don't need to get it right because the problems are not going to come about it. Hopefully there is something you can extract, which is why you talk about the Renaissance. We'll never go back to that state again." As Hans van der Meij (2003) documents, Carroll's Minimalism is still an actively pursued approach, and the focus of minimalism, namely extending on „people's prior knowledge, on their active construction of new knowledge and on their »learning by doing«" (ibid.) has created a wide diversity of different results.

3.2.5 A Differentiation from Folk Minimalism

Beauty is more than skin deep.
— popular saying

Although, given its roots in art history and aethetics, it is paradox to state that minimalism is also not an aesthetic dimension, this statement is true for a ‚folk' understanding of aesthetics: if aesthetics, αισθανομαι („perception by means of the senses") (Baumgarten, 1983, 79), are understood solely in terms of superficial appearance. Although the application of certain minimalist principles often results in a distinct appearance that has consequently been identified with minimalism, in all disciplines, minimalism has denoted more a perspective than a ‚look'. Minimalism is not simply monochromes in painting, it is not limited to atonality and absence of rhythm in music, nor is all minimalism in architecture connected to primary forms. Similarly, only a shallow transfer from typography would yield that minimalism is simply reduction of color and form. Minimalism is thus not just reduced design—doing away with too many colors, too many different typefaces, and too much screen clutter. And although simplicity is often a goal, minimalism cannot be equated with simplicity. Instead, as summarized in the final section of the first chapter, minimalism can be interpreted as different interpretations of ‚less is more', as a collection of principles that (while occasionally contradicting each other) go to extremes to create a focus of the recipient's—to increase ‚user' involvement, and as a way of looking at things—a minimal perspective.

Definitions for minimalism exist in abundance—a simple search of the Web delivers a wide range of definitions. Apart from those connected to art, music and literature, a shared understanding of the term—and a view often taken by interface and Web designers—seems to be connected to an aesthetics owing partly to minimalist typography: sparing use of colors, excessive white space, the absence of animation and visually distracting

content seem to suffice to classify a (web) design as minimalistic. According to Cooper (2003, 487), „the Web's history, in which many commercial Web sites began as marketing vehicles" is responsible for this backlash, „fancy graphics are not only distracting, but users have been well trained to ignore [...] marketing ploy[s] on the Web." (ibid.) The (now offline) collection of „minimalist web sites" collected by the „Minimalist Project"[6] lists example Web sites that that appeal to the need for more simple visual design. However, individual sites had little in common except for the generous use of white space and sans-serif fonts; the failure to capture similarities that go beyond the initial „look" of a site created a collection without creating a categorization; this could be one reason for the site's demise.

However, without conscious reference to the use of the term „minimalism" in other contexts, the informal, experience-based argument between different Web design blogs hints at the potential of minimalism as a concept: ‚dezwozhere' writes that „in short[,] a minimalist design may not always be what's needed[,] but a minimalist approach to design can be a powerful ally." (dezwozhere, 2003). And Olsen (2003) cautions his readers that „simplicity can be harmful", that a design aiming at visual simplicity can create structural complexities.

3.3 Defining Four Notions of the Minimal for Interaction Design

As minimalism is proposed as a tool to further the understanding of designing interactive systems, a first step is determining what aspects of the notion of minimalism in the fine arts are applicable to human-computer interaction, and what aspects cannot be transferred easily. It might be useful at this point to revisit the initial motivation to propose minimalism as a desirable alternative. This motivation starts with interactive systems that become too complex to be usable—McGrenere (2002) introduced the term *feature creep* for a frequent cause: the tendency of software products to accumulate more and more features, thus becoming more and more, and eventually too powerful. This in turn leads to *feature fatigue* (Thompson, Hamilton, & Rust, 2005) on behalf of the users: the goal is the design of simpler systems. As the inner complexity of interactive systems is, however, often caused by the complexity of the environment, designers cannot reduce inner complexity as they wish. While artists can reduce the statement of their work, and can fall back on enamel paint or the bare canvas to minimalize their art, designers must take into account the application context. Also, „industrial production" does not present an exception as it does in art, where the stroke of the painter was long taken as a measurement for quality—instead, „industrially" produced interactive systems must cope with a wide variety of situations.

This illustrates that a literal transfer of minimalism is of little use. What can be transferred is the mindset that stands behind employing minimal means, creating minimal structures, using minimal pattern, and including the recipient's interpretation in the ini-

6. The site is out of service. http://web.archive.org/web/20040930175437/textbased.com/minimalist/

tial design. For this thesis, these concepts define the minimalist standpoint[7] that is for HCI here defined in the following four notions of minimalism:

Functional minimalism denotes the reduction of the functionality of an interface. Ideally, only necessary core functionality is left in a functionally minimal design.

Structural minimalism refers to an access structure for an interface's functionality that is perceived to be minimal. In an ideal structurally minimal system, immediate support is lent by appropriate functionality.

Architectural minimalism focuses on reducing the complexity of an interface by transparently distributing its functionality across minimal parts. As an ideal result, the overall complexity is reduced without compromising the power of an architecturally minimal design.

C*ompositional minimalism* stresses the importance of the appropriation process for an interface. Paradoxically, compositionally minimal design must allow use patterns to develop after the design activity is over.

After an extended but necessary discussion of different interpretations of minimalism not covered in this thesis, defining what minimalism denotes here takes but a few words. The four notions that are presented above as a brief informal characterization of the underlying ideas, are in the following paragraphs individually deduced from characteristics of minimalism in the liberal arts and music, and set in relation to existing uses of the defining adjuncts before a more formal definition is given. What follows is a discussion of the possible interaction between the different notions of minimalism—based on their roots in the other disciplines.

3.3.1 Minimal Functionality for User Interfaces

As mentioned in the previous chapter, painting in minimalism tried to further reduce modern painting. This was achieved in two distinct ways, that of reducing the means (2.4.1), and that of reducing the meaning (2.4.2). While the reduction of means focused on the artist, his actions, and on the concept of artistry, the reduction of meaning used similar techniques, but focused on the *function* of painting. Once the depiction of reality was devalued by photography, abstract painting focused on depicting the unreal—and carried it to an extreme: painting was considered to be nothing but what is visible; color and form stood only for themselves. Thus, painting became devoid of function; in Reinhardt's words, it became „art-as-art" (Reinhardt, 1975, 53).

Although some interaction designers go to similar extremes and design objects that are without a function, more often, the extreme minimum of functionality is a singular func-

7. The term *standpoint* was chosen here to distinguish the individual minimalist perspective that relates to one of the four notions of minimalism from the stance that all four perspectives share. Although minimalism lacks the social orientation that is customarily associated with a standpoint (comp. SEP, 2003), it shares the claim of epistemic privilege over what construes reduction, and simplicity.

tion. The idea underlying this *functional minimalism* is grounded in a common observation: a good tool can often *only* do one thing, yet do this one thing very well. The concentration on a single purpose allows it to be adapted very well to the task, trading off flexibility. The focus on the whole is translated as a focused competency of the tool: as the tool's functionality is clearly defined, all energies can be concentrated on perfecting its performance. The construction of such tools is common in many traditional crafts; an example for classes of such tools would be the variety of hammers and drills a woodworker chooses according to the task.

Between interactive systems and manual tools, however, there is a difference in complexity: while the construction process of an analog tool, such as a hammer or drill, may be based on complex testing and accumulated experience, and the production process can be very elaborate (e.g. for damascene steel), the resulting tool itself is often not complex. Interactive systems, however, are almost never internally simple, as even the operating systems needed as a basis is a most complex construct. Thus, the focus of *functional minimalism* is not on building simple tools in a simple way, it is about perceiving tools as functionally minimal—i.e. as useful for only a single purpose. As perceived functionality is not a reliable measure, the stricter notion of *accessible functionality* takes its place. A functionally minimal tool (e.g. a mobile phone) can have complex functionality (e.g. accessing radio cells, choosing a provider, keeping track of battery life) that—as long as it is never visible to the user—does not decrease its functional minimality.

> *Functional Minimalism* refers to a reduction in *accessible functionality*. An interactive system is increasingly $\boldsymbol{minimal}_F$ as it presents fewer features to the user. A $\boldsymbol{minimal}_F$ system is identified by its focused competency.

Definition 3.5: Functional Minimalism.

3.3.2 Minimal Structure for User Interfaces

It is very tempting to take functional minimalism as the most important quality that minimal products can possibly have. If the purpose of a device is both singular, and clearly defined, the ideal consequence is that the device simply functions—without navigational overhead. However, examples that come close are either replacing technology with intelligence, e.g. in the beginning of the phone era, where operators did all the work, or devices that must operate in emergency situations, where operational mistakes can be fatal, or high stress is expected (e.g. elevator intercoms tend to have few buttons). Few technical designs are that simple in terms of functionality.

Instead, complexity could almost be said to be a hallmark of digital designs. As computers are understood as universal machines, there is obviously little that they can't do, and less that products don't promise to do. Striving for functional simplicity directly conflicts with the users' call for power that is amplified by marketing (feature comparison) and echoed in research and development (feature addition). Thus, an often sought solution is to create not a product that is actually minimal in features, but to change its *perception* towards the minimal.

One aspect of minimalist painting is that it tried to reduce the function of the artwork; consequently, the inner structure was minimized. While this compares to functional minimalism, structural minimalism is more similar to another perspective inherent in minimalism: other minimalist art only tried to seem simple—either through making its structure transparent, or by cleverly hiding complexity (2.4.3). The latter approach is mirrored in digital designs—here, complexity is commonly hidden through layered *access* to features—functionality that is generally used more often, or in specific, system-determinable situations, is made directly accessible; if necessary, this involves a trade-off where less immediately necessary functionality is made more difficult to access. Structural minimalism describes this reduction of structure in designs—and focuses the meaning of structure to the *structural access to functionality*. By contrast, computer science usually uses structure in a broader sense, and includes the invisible inner structure; within the context of this thesis, inner structure that remains invisible is of no interest, and inner structure that defines the interface is covered by the notion of architectural minimalism below.

> *Structural Minimalism* refers to a reduction in *perceived access structure*—in some cases, this is equivalent to a reduction in *visible functionality*. An interactive system is increasingly ***minimal***$_S$ as it confronts the user with less navigational structure. A ***minimal***$_S$ system is discerned by its optimal fit of provided and necessary functionality—at any point in time.

Definition 3.6: Structural Minimalism.

3.3.3 Minimal Architecture for User Interfaces

Taking seriously the idea that interaction design reaches beyond designing pretty (useful) interfaces, architectural minimalism takes a more technical perspective to the adaptation of functionality to use situations, and focuses on the activity of *building*, or on the resulting *construction* of the interactive system. The etymology of architecture leaves wide room for interpretations of the term: the greek „master builder" (a compound of αρχι, „chief", and τεκτων, „builder, carpenter") has in modern architecture developed into a designer, engineer, and facilitator (comp. Kostof, 1977, 324; Straus & Doyle, 1978). Architecture is interpreted as „the art of the articulation of spaces" (Zevi, 1957), and although architecture lacks the self-referential and self-modifying power of language (Johnson, 1994, 423), it shares with language the quality of communication; „the public can respond to the explicit metaphors and messages of the sculptors" (Jencks, 1977, 6).

This wide spectrum of meanings is echoed in the current use of the term architecture in computer science, where it is mainly referred to in software engineering: Architecture is used to describe both the (social) process (comp. e.g. Coplien, 1999) and the product of conscious structuring activities aiming to improve the legibility, maintainability, and reuseability of source code (e.g. Bass, Clements, & Kazman, 1998). As software engineering focuses on the engineering process used to manufacture complex software systems, architecture defines in this scope the internal qualities of a software, and need not be visible to the end user at all (Floyd & Oberquelle, 2003).

Within the context of this thesis, the use of architecture is not limited to the domain of implementation; instead, based on the focus on user interface design, *minimal architecture* refers to qualities of building and construction that are still visible in the design and can be experienced by the end user. This use of architecture is more closely related to another use of the term in computer science: the notion of information architecture, which is used to describe the practice of structuring information (for the World Wide Web). Its definitions tend extreme generality; Wurman (1996) defined of information architecture as „the merging of three fields: technology, graphic design, and writing/journalism"; due to these different perspectives, collaborative definitions of information architecture add little detail, e.g. „Information Architecture is inherent, practical or theoretical knowledge having to do with the presentation of written, spoken, graphical or other information." (IAwiki, 2006). By contrast, architectural minimalism puts emphasis only on the reductive qualities of ‚architectural practice' in design.

Based on the characteristics of minimalism identified in the previous chapter, the architectural perspective on the interfaces of interactive systems focuses on how necessary complexity can be achieved with simple elements that are combined transparently for the end user—as elements of a music piece or of a piece of artwork can be simple in themselves, yet complex in combination (2.4.4). An architecturally minimal system groups functionality according to its use; unlike the perspective defined by structural minimalism, architectural minimalism starts beneath the superficial understanding of interface: minimal building blocks are combined as simple, identifiable tools—just like the reduced form of the square used by minimalist artists—to form a more complex system. Acknowledging the fact that few information processing tasks can be solved by appliances, finding small, comprehensible or at least recognizable parts within the system helps the user to understand system reactions and correctly predict system behavior.

> *Architectural Minimalism* refers to a reduction of perceived complexity by *externally visible distribution of responsibility* (in the sense of „partitioning of functionality"). An interactive system is increasingly ***minimal***$_A$ as it provides combinable blocks of coherent functionality to the user. A ***minimal***$_A$ system is typically not adapted to a specific task. Through combined use of its parts, necessary functionality can be combined as necessary.

Definition 3.7: Architectural Minimalism.

3.3.4 Minimal Composition for User Interfaces

The design of interactive systems is often based on predictions about their use—and taking them for truth, will often determine exactly how they are to be used, and consequently, how tasks have to flow, how work is to be performed. Although a functional specification, and thus implementation, is aided by clear and well-defined sequences of activities, from the perspective of human-computer interaction, the trade-off in flexibility can overshadow the technical benefits.

In art, many artists are very decided on what their works describe, and what they mean. Some minimalist artists took a different stance: the spectator was included in the work as the meaning unfolded only with the viewing, and the personal experience was valued above the artists private conception. In minimal music, the „formal specification" of music was often considered less valuable than the experience creating by performing a piece with other musicians (2.4.5).

Compositional minimalism stresses that the performance is more important than the composition—it proposes that composition should be minimal in order to allow a maximum of interpretation. As minimalist painters stressed that the value of their pictures lies within the spectator, the value of an interactive system does not exist without its users. For a tool to be optimally useful, it must do the user's bidding, not the other way around. However, even when a tool is designed with minimal functionality, and more so when it is designed for a single task, unnecessary constraints might be introduced. Like the minimalist composers, designers should carefully consider where one should refrain from assuming use contexts and well-defined sequences of actions to be static. If it is possible to limit the design to minimal use patterns that can be appropriated in use, the result is a more flexible design. This does not mean that tools should not be conceived for a very specific purpose—examples for successful adoption for other tasks will be given. Instead, the modularity of a system should not only be considered on an infrastructure level, but also on a task level: systems should be designed in a way that their building blocks can be combined in new ways to master new use situations.

> *Compositional Minimalism* refers to a reduction of aspects decreasing the tool's usefulness for other tasks through *specificity for planned tasks*. An interactive system is increasingly ***minimal***$_C$ as it makes fewer assumptions about its use and places less restrictions upon its users. A ***minimal***$_C$ system encourages its use in contexts and tasks not considered during design.

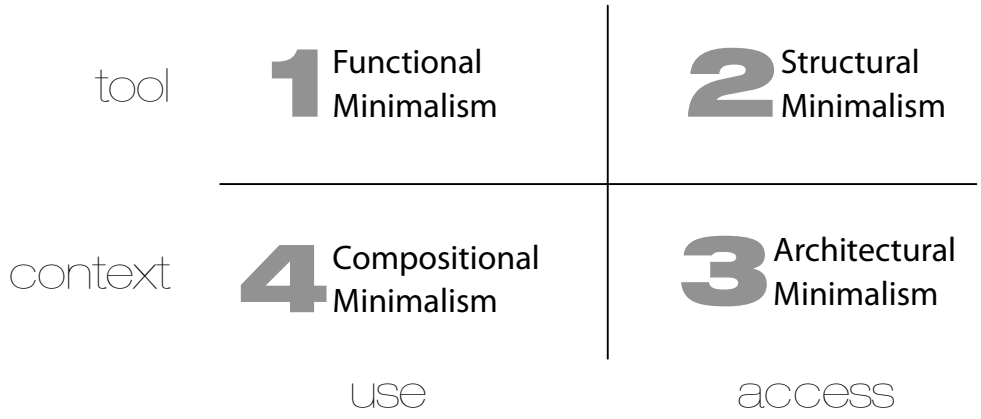
Definition 3.8: Compositional Minimalism.

3.3.5 A Minimalist Terminology for the Design of Interactive Systems

In the previous sections, differences between the four notions of minimalism put forward here have been marked out. All four notions of minimalism describe aspects in which the design of a *user interface* for an interactive system can be considered minimal, and although they are each based on a different understanding on what should be minimized, some overlap, or at least strong interactions between the different notions can be observed. To illustrate this, the four notions of minimalism have been laid out as four sectors (Table 2): Functional minimalism and structural minimalism both affect the system's *user interface* directly, while architectural and compositional minimalism work at different levels, namely the inner construction of the system and the users' ‚interface' to the system, or the integration with other tools. Structural and architectural minimalism both focus on adapting the tool to the task, modifying the structure by which functionality is ac-

cessed, while functional and compositional minimalism stress the use of the tool and the context of use.

Table 2: Arrangement in four fields



Each of the four notions of minimalism defines a different perspective on an interaction design; although different designs will exhibit properties that match these notions to different degrees, it will be difficult to impossible to find practical examples demonstrating to a single notion of minimalism. How much conceptual overlap exists is best demonstrated in the practical application of the notions of minimalism for analysis (chapter 6). Here, some of the manifold interactions between the different notions of minimalism are anticipated:

(1–2) Functional and structural minimalism: Minimal functionality can be easily provided using a minimal structure—fewer functions are less difficult to structure. At first glance, a differentiation between functional and structural minimalism might thus not seem useful. However, although a truly minimal functionality (a single function) can map naturally to a minimal access structure (a single button), the reverse is not necessarily given. As the example of the Morse key (3.2.1) demonstrated, the two notions of minimalism are almost perpendicular: a minimal access structure can be created for complex functionality. And even for a minimal functionality, a complex access structure can be constructed by introducing many options that concern the status of the tool instead of the task. The minimality of an access structure—its „optimal fit of provided and necessary functionality" (def. 3.6)—becomes effective only if a prioritization of functionality is possible. Alan Kay coined the proverb „Simple things should be simple. Complex things should be possible" on this subject (Smith, Irby, Kimball, & Verplank, 1982).

(2–3) Structural and architectural minimalism: Both structural and architectural minimalism affect the perceived access structure to functionality. The difference given in the definitions is that structural minimalism concerns primarily the superficial structure that is overlaid on functionality to make it accessible, while architectural minimalism also denotes the internal construction of an interactive system; it suggests distributing functionality over building blocks that transparently form the overall design. A challenge for the latter part of this thesis is thus to test whether this discrimination yields practical value

analyzing and constructing interactive systems. Often in HCI, the difference between superficial decoration (german „Oberflächengestaltung"), access structure (information design) and system construction is blurred; however, the tendency to prefer early intervention by usability methods is clear and partly stems from the desire to reorganize work rather than screen layout (def. interaction design, CD, etc.). Architectural minimalism as defined here minimizes the access structure of a design, as a clear division of work between individual tools (i) allows the user to choose his tools, and (ii) makes the individual tools functionally simpler.

(3–4) Architectural and compositional minimalism: As the summary of the use of building blocks in minimalism suggests (2.4.4), an architecturally minimal system will enable interpretation by the user, and thus further the system's qualities that are referred to by compositional minimalism. Common to both architectural and compositional minimalism is a design perspective that includes aspects going beyond the single product. Architectural minimalism tries to partition functionality within a single application or environment into interoperable modules. Compositional minimalism tries to extend the usefulness of a design beyond a single application or environment; much of the still emerging field of ubiquitous computing that interacts with stationary computing systems applies a mixture of both minimalisms (e.g. include post-it notes, mackay). Another, more technical area where architectural and compositional minimalism overlap is the development of open standards for data interoperability (e.g. in the World Wide Web). The minimalisms differ in perspective; while architectural minimalism explicitly considers the whole functionality of a design, compositional minimalism focuses on the use of often only partial functionality within a larger context.

(4–1) Compositional and functional minimalism: To demonstrate differences between compositional and functional minimalism, the analogy of the simple craftsmen's tools that make a skillful orchestration of different tools necessary must be extended as in this example, compositional and functional minimalism seem to fall together. While functional minimalism often creates the need for more tools—and thus the basis for an orchestrated activity, compositional minimalism is possible using very complex tools (e.g. a combination of spreadsheet, statistics, drawing, and text-processing software is often used to generate scientific illustrations, engaging only a tiny subset of functionality of the complex systems involved in the process). Compositional minimalism is thus about interoperability and about enabling the user to orchestrate use of component working together to achieve the set goal—different tools within a software, different applications within a system, or even different systems within a network are used to implement her instructions.

The relationship of notions of minimalism that are arranged in adjacent fields in Table 2 is comparatively close, and some conceptual overlap exists. Looking at the more remotely connected notions of minimalism, the differences become more clear:

(1–3) Functional and architectural minimalism: Both functional and architectural minimalism reduce functionality; the scope of reduction in functional minimalism is the total de-

sign; in architectural functionality, functionality is reduced by defining modules that encapsulate functionality that is similar with regard to the task. The resulting partition of functionality remains visible in the user interface—their appearance can range from ‚sub-applications' to ‚inspector tools'. A functionally minimal system is by definition difficult to split into several parts, and architecturally minimal systems are designed to allow complex interactions of their building blocks—thus, they are often not functionally minimal. Functional and architectural minimalism can thus be considered *alternative approaches* to reduction.

(2–4) Structural and compositional minimalism: Making the life of users easier is the purpose of both structural and compositional minimalism; both try to give the user mmore freedom, structural minimalism by freeing her from unnecessary navigation, compositional minimalism by removing unnecessary workflows that might contradict her action plans. However, following a paradigm of structural minimalism can effect a more restrictive composition; as the access structure is increasingly tailored for one task, it assumes a standard workflow that will help those following it, but make ‚creative misuse', or appropriation (cf. Pipek, 2005) more difficult.

3.4 Summary

This chapter defined four notions of minimalism that apply to the design of interactive systems. The four notions draw on aspects of minimalism that were identified in the last chapter as being shared by art, music and other disciplines (2.4). Each notion identifies an individual perspective on design, and suggests a certain form of reduction.

The four fields for reduction are the *functionality*, the *structure*, the *architecture*, and the *composition* of a design. Together, these four different interpretations of minimalism form a minimalist standpoint that goes beyond simplicity[8] (3.2.1) and differs from existing concepts that are used by computer science (3.2.2–3.2.5).

The definition of four different notions of minimalism (3.3.1–3.3.4) does not imply that a designer can assume one of the presented perspectives, and reduce without influencing other aspects of her design. Instead, some conceptual overlap, and interactions between the different notions of minimalism are made explicit (3.3.5).

8. Minimalism is introduced in this thesis as a conceptual tool to distinguish different types of simplicity. Reduction is used as a means to achieve simplicity. Different approaches to reduction can be differentiated using the notion of simplicity, e.g. returning to the wire telegraphs, the dial telegraph is structurally minimal, while the lever telegraph is functionally minimal, making it a more universal tool that can be easily misused, or linked to other tools (e.g. the structurally minimal Morse code, or machines that are much faster at creating the binary signal transmitted)—it is more minimal in both composition and architecture.

4 Minimalism and Familiar Perspectives

This thesis tries to establish minimalism as a novel perspective for the analysis of design for interactive systems. The innovative merit of the different notions of minimalism lies in their focusing function: *functional*, *structural*, *architectural* and *compositional minimalism* highlight different *reductive tendencies* of a design. In chapters 5 and 6, their function is to identify simple aspects of existing designs, and how new and existing design techniques can be used to create simplicity. The purpose of this chapter is to examine how the minimal standpoint differs from established points of view (Figure 35).

This chapter begins with a historical look at industrial design as the forerunner of HCI as a discipline. The relationship of minimalism and design principles represented by international standards for usability illustrates a focus shift from individual functionality to a larger context. Minimalism as a perspective for existing HCI lore is examined using popular guidelines as an example, and advice resulting from longtime individual experiences of renowned HCI experts that illustrates the importance of reduction in design.

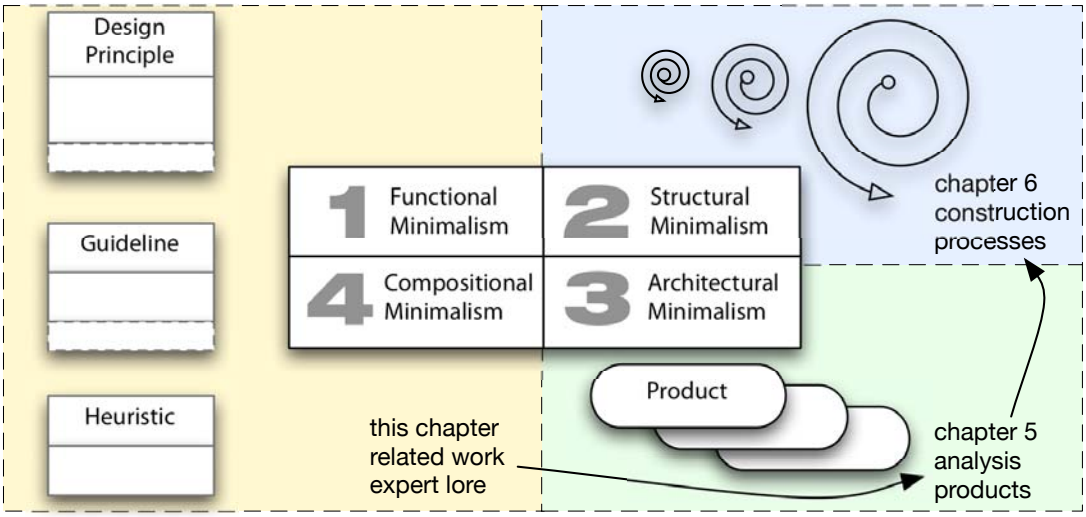


Figure 35: Minimalism connects values, products and procedures in this and the following chapters.

4.1 HCI, Design and Minimalism

> It must be borne in mind that the object being worked on is going to be ridden in, sat upon, looked at, talked into, activated, operated, or in some way used by people individually or en masse. If the point of contact between the product and people becomes a point of friction, then the designer has failed. If, on the other hand, people are made safer, more comfortable, more desirous of purchase, more efficient—or just plain happier—by contact with the product, then the designer has succeeded.
>
> —Henry Dreyfuss, Harvard Business Review, November 1950

While computer scientists often consider themselves as part of an engineering tradition (e.g. Jehn, Rine, & Sondak, 1978), the part of human-computer interaction that tries to gather contextual knowledge and transfer that into the design of an interactive system is heavily influenced by industrial design (Sloan, 1975; McCrickard, Chewar, & Somervell, 2004; Wolf, Rode, Sussman, & Kellogg, 2006). Not only do product designers contribute to the field of human-computer interaction—visible in conferences series such as *ACM Designing Interactive Systems* and special tracks of most HCI conferences—but industrial design can be considered an important predecessor to the design of interactive systems.

Today, when we design a computer artifact, when we build an operating system, develop a service, or design an application we are directly changing the organization of work, and thus impacting the lives of people. Industrial design became conscious of this responsibility before the widespread use of computing technology, and can in many aspects be called a precursor to the discipline of human-centered design. In early industrial design, experiences from designing for audiences, e.g. from designing theatre stages, were transferred to the design of mass produced technology.

Industrial designers had to mediate between stakeholders—much as usability experts today, they had to find compromises that would make CEOs, engineering and marketing departments equally happy—and find useful designs for the prospective users. As the role of the designer has not changed as much as one might believe, it is not surprising that many of the tools applied in interaction design were already in the hands of industrial designers: Mock-ups were widely used, focus groups were common, and even a form of personas was employed to communicate target groups to the design team—although more in form of generalized measurement for ergonomic designs (Dreyfuss, 1955, 69f).

Industrial design is a profession formed during the 1950s when the shift from manufacture to mass production allowed a before unheard-of abstraction in the design of everyday products (Bürdek, 1991; Bürdek, 2005) where one successful design could reach millions of households—instead of only few customers of the traditional craftsman. The new form of production led to a different approach to thinking about design; since many different users would be targeted by the design, the functional aspect was stressed (Dreyfuss, 1955). This replaced the focus on decoration that before had identified the individual craftsman who created a product.

Dreyfuss gives an example for this new type of design, created when „the manufacturers of the Mason jars sought a change. Here was a temptation to prettify a familiar object. Happily, we resisted it. We made the jar square with tapered sides, thereby … saving storage space, providing a jar that wouldn't roll off the table, and handing the manufacturer a thumping sales advantage. It was another lesson in the basic simplicity of good design" (ibid., 78). While some of his contemporary designers enjoyed the ubiquitous streamlining that started with vehicles, and did not spare the breakfast table, he noted „call it cleanlining instead of streamlining, and you have an ideal that the designer today still tries to achieve" (ibid., 77).

Figure 36: American Airstream caravan (1964), streamlined to an idealized „teardrop shape".

Industrial design tried to find an aesthetic form that was both pleasing and functionality, sometimes sacrificing originality for functionality: „If something is to be lifted or operated by a handle, we try to integrate the lifting device into the design, but never to conceal it. At the expense of forfeiting originality, and it is a great temptation to hide locks and access panels, we try to make things obvious to operate" (ibid., 71). This did not lead designers to ignore other qualities—as Dreyfuss recalls, „One time, I arranged to get behind the counter of a drugstore to catch reactions to a new … clock we had designed. My first customer was a woman, and I showed her our model and a competitive clock of the same price. I watched her weigh a clock in each hand. I was confident in her choice, for we and our client's engineers had labored long and hard to make out clock light, believing lightness was an expression of its excellence. I had a sinking feeling as she bought the heavier clock. But it brought home the lesson, that to some people weight can be a sign of quality, also that the designer must appreciate that some things demand weight and some lightness, and he must determine when each is a virtue." (ibid., 68). This point is taken up by Don Norman (2004c), who criticizes pure functionalism: „If something is to be lifted or operated by a handle, we try to integrate the lifting device into the design, but never to conceal it … At the expense of originality … we try to make things obvious to operate."

Figure 37: Philipp Starck: Lemon press „Juicy Salif" (1990).

In the same collection of ideas, Norman argues that rich and complex objects can hold our attention longer, giving the example of the ever new experience of listening to richly textured classical music (ibid., 109f). When design becomes purely emotional, however, it ceases to create tools. An example Norman (ibid., 112ff) gives is the lemon press designed by Philippe Starck (Figure 37) that looks great, but poorly performs when one tries to actually use it. Norman, by the way, ‚admits' to own a gold-plated version that, naturally, should never come in contact with acids.

The emphasis on beauty as an end in itself is also contradicted by Christopher Alexander's Zen View, who advises that the transient is more attractive: „if there is a beautiful view, don't spoil it by building huge windows that gape incessantly at it. Instead, put the windows which look onto the view at places of transition—along paths, in hallways, in entry ways, on stairs, between rooms." (Alexander, 1977, pattern 134)

Among designers, preferences are not distributed evenly. While there is a long tradition of functionalism, and industrial design anticipated many of the practices of interaction design, designers can also simply be interested in beauty for itself. Following Gelernter's (Gelernter, 1999) perception of „machine beauty", this thesis limits itself to discuss those aspects of design aiming at the creation of powerful tools. Vice versa, one can conclude that the reflective notion of minimalism is useful only for some designs as it inherently focuses on the tool-qualities of a design.

4.2 Standards in Interaction Design and Minimalism

For those in doubt about the meaning of usability, and about the goals interaction design strives to achieve, there is an international answer: formal standards of the International Standards Organization (ISO) represent an agreement between international experts; for the sake of clarity, these standards tend to define desirable values for software (Bevan, 2001).

Minimalism is proposed here as a perspective on the proposed values that focusses on a distinctive set of similarities, and thus promises to highlight some implicit connections

between the values, and between the values and the methods discussed later. Even in this informal format, this is a useful complement to a direct match of evaluation methods and values: HCI tends to place less weight on the design aspect, which in turn makes evaluation a dulled tool—the evaluated designs do not specifically aim for the tested qualities, and constructive rather than comparative criticism becomes difficult. As will be illustrated briefly, quantitative evaluation is also incapable of capturing the more complex connotations created between the values by the minimalist perspective. Minimalism is put forward here to explain why the values described here—and later, why existing guidelines and methods—can lead to better design results. The reflection on the values set by HCI standards is thus a first test whether minimalism can be used successfully to assess the design of interactive systems; as a further benefit, already this first application adds to the specification of what is understood here by minimalism.

While a multitude of different standards exist that formalize requirements for hardware or software interfaces, or the development process (comp. Bevan, 1995), only a single standard international standard exists that aims to define general measures and values of usable systems: *ISO 9241*, now retitled „Ergonomics of Human System Interaction", defines usability as a measurable quality of a product based on effectiveness, efficiency and user satisfaction in part 11 (see definition 4.1).

> ***Usability***
> The *effectiveness*, *efficiency*, and *satisfaction* with which specified users achieve specified goals in particular environments.
> ***Effectiveness***
> The *accuracy* and *completeness* with which specified users can achieve specified goals in particular environments.
> ***Efficiency***
> The *resources expended* in relation to the accuracy and completeness of goals achieved.
> ***Satisfaction***
> The *comfort* and *acceptability* of the work system to its users and other people affected by its use.

Definition 4.1: Measurements for the use quality of software according to ISO 9241/11.

ISO 9241-11 thus defines usability as effectiveness, efficiency and satisfaction. All three can be measured quantitatively using pre-defined tasks, time measurements and questionnaires for user satisfaction. The measures do, however, not attempt to explain why a software is more usable, or even more effective, efficient or satisfactory. To this end, ISO 9241 further defines a number of qualities for usable software. While software is usually examined for compliance with ‚inner qualities' (Floyd & Oberquelle, 2003)—software should be correct, easily understood and changed—in ISO 9241/10, the ‚outer quality', or use quality of software is defined in the following terms:

Suitability for the task
A dialog is suitable for the task if the dialog helps the user to complete his task in an effective and efficient manner.
Conformity with user expectations
A dialog is conforming with user expectations if it is consistent and complies to the characteristics of the human user, that is his knowledge of the task, his education and expertise and generally accepted conventions).
Self descriptiveness
A dialog is self descriptive if every single dialog step can immediately be understood by the user based on the information displayed by the system or if there is a mechanism to obtain any additional explanatory information on request of the user.
Controllability
A dialog is controllable if the user can start the dialog and influence the direction and speed of the dialog at any point in time until the task has been completed.
Error tolerance
A dialog is fault tolerant if a task can be completed with erroneous inputs with minimal overhead for corrections by the human user.
Suitability for learning
A dialog is suitable for learning if it helps and guides the novice user to learn about other perhaps more efficient or effective ways to use an application.
Suitability for individualization
A dialog is suitable for individualization if the system allows to adapt the interaction style to a particular task and the specific capabilities and preferences of the user.

Definition 4.2: Use qualities of software according to ISO 9241/10.

These use qualities universally define how usable designs should behave. The relationship between the seven qualities mentioned in ISO 9241/11 and the four notions of minimalism can be subsumed as follows:

Functional minimalism has two immediate consequences for the use qualities of a design: fewer functions mean that there is less to learn—which will often result in encouraging the user to better apply a minimal tool, and that control of the the less complex functionality is supposedly easier—complexity often causes the users' mental models to fail predicting system responses. There is also a shift in the meaning of suitability for the task: this is often interpreted as effectiveness, meaning „will do anything that the task requires". Instead, for functionally minimal designs, the question of ‚*toolness*', or „will do something that the task requires" becomes more central. Functional minimalism could contribute to self-descriptiveness, but it might harm error tolerance in the sense that understanding er-

rors can become more difficult when they are not produced by a single tool, but through the interaction of different tools.

Structural Minimalism means fitting the access structure of the system to the tasks that will be executed. Generally speaking, often used functionality should be readily available, at the cost of less often used functionality which may be accessible, but more difficult to invoke. This can be understood as a fit created either by explicit design, or by customization—customization of interactive systems often begins with users changing access structures such as menus or toolbars to improve the accessibility of often used functionality. Two different approaches to structuring functionality can be distinguished: If the design of a structurally minimal system groups functionality not only according to frequency, but also into meaningful units, discovery of functionality, and thus learning is supported. This can involve a trade-off as consistency with other applications is reduced, and initial learning hindered. The definition of structural minimalism is based on the „minimal perceived structure"—which is partly congruent with conformity with user expectations: when user expectations regarding which functionality should be invoked are met, navigation in e.g. menus is reduced. A structurally minimal system can be easier to control—but if it is structured only based on frequency, users might have problems getting an overview, and actions the designers did not plan for can decrease the users' sense of control.

The partitioning of functionality that characterizes *Architectural Minimalism* results in a transparent distribution of responsibility for task activities across the tools a system is composed of. This contributes to the suitability for learning as tools can be discovered and mastered individually. As users can simply choose which tools—and thus which part of functionality—to use, it shifts the meaning of suitability for individualization from interface customization to appropriation; the interface is not changed, but instead used with a different focus. Conformity with user expectations is supported by the grouping of functionality into visible tools; if users share the designer's understanding about the utilization of these tools, conformity is established on the tool level, and the location of individual functionality becomes more predictable.

Finally, *Compositional Minimalism* continues the change of focus introduced by the preceding notions of minimalism: the interactive system and its design become less important than its actual use. Qualities that ISO 9241 considers to be attributes of the software become both requirement for and result of a design. For example, the conformity with user expectations on the tool level is necessary to enable appropriation. Appropriation then results in conformity on the system level that was not determined by the initial design. Individualization partly corresponds to compositional minimalism—a compositionally minimal system tries to allow different individual uses. Yet, these uses do not always result in a changed tool. Similarly, the focus in learning shifts from discovering new functionality to discovering new uses of known functionality.

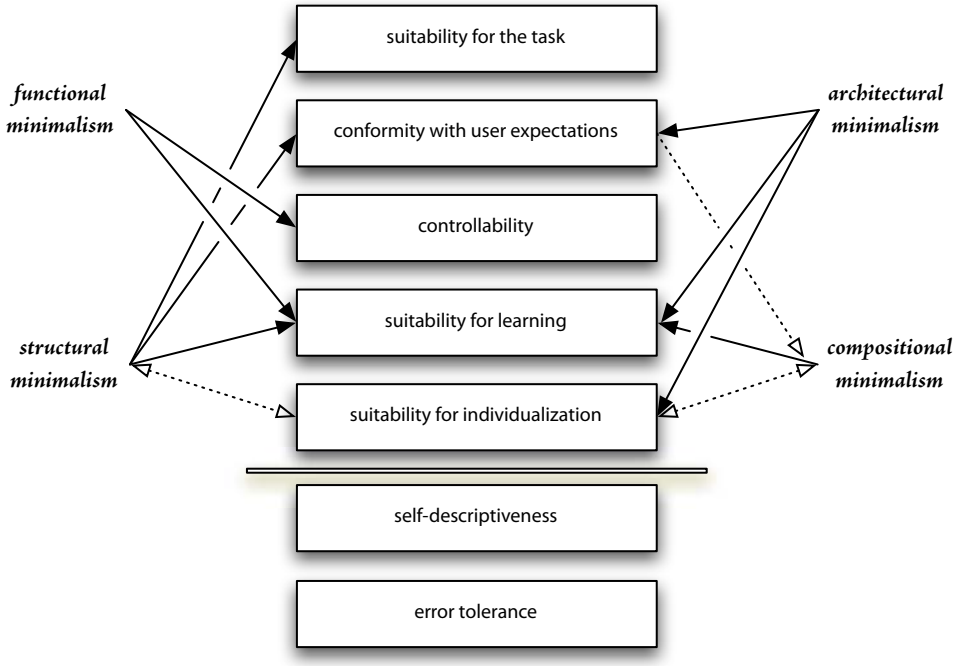


Figure 38: Notions of minimalism and the use qualities of ISO 9241/11.

The perspectives created by the four different notions of minimalism demonstrate an understanding of usability that relates to the standard, yet changes the viewpoint of analysis in fundamental ways:

(1) *Functional*, *structural*, *architectural* and *compositional* minimalism focus on *qualities of a design* that further its use qualities. The relationship cannot be mapped directly as several aspects of minimality influence e.g. the use of learning. Minimalism, however, provides a more constructive that reflective perspective, allowing prediction of use qualities.

(2) The scope of *use* and *learning* is extended beyond the single system. An interactive system becomes a tool among other tools. Instead of *use qualities* of an interactive system, the interaction with work processes, other tools, and other users becomes central.

Carroll (1990, 322) notes that „an important body of work in current HCI stresses that designed artifacts cannot be understood apart from the situations in which they are used." Minimalism further implies that not only the situation, but also other designs used situatively need to be included in design considerations. The all-present opportunities for interactions with other designs is stressed by the important theme of ubiquitous computing (Weiser, 1991) in HCI.

As a consequence of this broadened perspective, evaluation becomes inherently more difficult. While usability metrics have been developed (comp. Dix, Finlay, & Abowd, 1997) to measure the standard use qualities of software, these are either determined by

what can be measured quantitatively, or necessarily indirect and relative as they measure the interpretation of users (compare Table 3).

Table 3: Methods for measuring usability according to ISO 9241/10. Excerpt from Dix et al. 1997.

Usability Objective	Effectiveness Measures	Efficiency Measures	Satisfaction Measures
Suitability for the Task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of "power features" used	Relative efficiency compared with an expert user	Rating scale for satisfaction with "power features"
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for "ease of learning"
Error Tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

The definition of these usability metrics demonstrates that evaluation is often focused on features, not on tasks. Functions of the design are no longer tools to accomplish tasks, but their number is used as a benchmark for software quality itself, e.g. when learnability is measured as learned features set in relation to total features. Measuring the minimality of a design simply by asking users promises limited success as it is difficult to separate the aesthetic and the functional perspective in user surveys (5.5.1). Users may also not care about minimality as they do about learnability or suitability for the task (5.1.3).

The minimal perspective, however, demonstrates limitations of the usability metrics: Comparing a functionally minimal design with a design providing more functions highlights the importance of task design for measured user satisfaction—often, task design is oriented on collected requirements, and thus on a tool's functionality, rather than on real user needs. Satisfaction is an inexact measure as it is the result of complex interactions and cannot be linked unambiguously to a single use quality. Usability metrics focus on the designed system alone, e.g. by observing the use of its „power features". By contrast, the notion of compositional minimalism indicates that instead of examining the system in isolation, intelligent appropriation of tools within actual work is more relevant for the usability; it cannot, however, easily be measured in laboratory environments.

From the standpoint of minimalism, the use qualities defined by ISO 9241 seem limited in their ability to guide and evaluate usability. Karat & Karat (2003) agree: „The importance of these two documents lies not as much in the specifics of the guidelines included in the various parts as it does in the acknowledgment that usability is complex and context-dependent. Designing for usability should be seen as involving both attendance to high-level principles (such as those in Part 10) and selection of particular approaches that are determined by a context of use that is broader than we have generally attended to."

4.3 HCI Lore and Minimalism

After international standards, the most important guides for usability are collections of heuristics that describe ‚good practices', or qualities of software in more concrete terms than ISO standards. In the next two sections, ‚Golden' and ‚Silver' rules for design are linked to minimalism, as are interface guidelines that exist for most operating systems.

4.3.1 Rules of Noble Metal and Minimalism

The standards provide an authoritative source of reference, but designers without usability experience have great difficulty applying these types of guidelines (Souza, & Bevan, 1990). The services of renowned usability experts are thus in high demand—yet these are not always available. Experience gathered over the course of many design projects is what usually distinguishes an expert. Often, the resulting how-to knowledge is communicated to less experienced designers by means of „rules". These guidelines are often used in practice, as they are concise, often catchy, and the nature of their source—respected members of the community—gives them great authority.

Following the realization that graphical user interfaces are not automatically „easy to use" (Zanino, Agarwal, & Prasad, 1994), rules were contrived to guide design and evaluation. The first set of rules that became well-known are the *Eight Golden Rules* that were coined in 1987 by Ben Shneiderman (Shneiderman, 1987). Yet, in his latest recollection of these rules, Shneiderman (2004, 63) acknowledges previous work that he built upon. An example are Smith and Mosier's (1986) data display guidelines; they present five high-level goals for data display and data entry each (Table 4).

Table 4: High-level goals for data display and data entry (Smith and Mosier, 1986)

1. Consistency of data display	1. Consistency of data entry transactions
2. Efficient information assimilation by the user	2. Minimal input actions by user
3. Minimal memory load on the user	3. Minimal memory load on users
4. Compatibility of data display with data entry	4. Compatibility of data entry with data display
5. Flexibility for user control of data display	5. Flexibility for user control of data display

The second and third goal immediately link to structural minimalism: Smith and Mosier require the access structure to information and functionality to be as minimal as possible. They also stress as a final goal the flexibility of use. While their scope is only a single dialog, compositional minimalism builds upon this flexibility. Both the first and fourth goals concern internal consistency—this important aspect is discussed below (4.6).

Shneiderman's *Eight Golden Rules of Interface Design* (Shneiderman, 1987; Shneiderman, & Plaisant, 2004), reproduced in Table 5, present a different focus. These rules, much as those formulated by Molich and Nielsen (Molich & Nielsen, 1990), result from testing the usability of different designs—Molich and Nielsen note that „This checklist reflects our personal experience". Molich and Nielsen's *Silver Rules* were coined to guide heuristic evaluation, Shneiderman gives advice to designers. Both sets of rules, however, try to im-

prove a design by preventing common usability mistakes. Thus, they focus less on qualities of a design, but rather present advice in form of principles.

Table 5: Excerpts from Shneiderman's Eight Golden Rules of Interface Design.

1. *Strive for consistency*. … consistent sequences of actions should be required in similar situations; identical terminology should be used … consistent color, layout … should be employed throughout. Exceptions … should be comprehensible and limited in number.
2. *Cater to universal usability*. … design for *plasticity*, facilitating transformation of content. Novice-expert differences, age ranges, disabilities, and technology diversity each enrich the spectrum of requirements that guide design. …
3. *Offer informative feedback*. For every user action, there should be system feedback. …
4. *Design dialogs to yield closure*. Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group give operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans from their minds, and a signal to prepare for the next group of actions. …
5. *Prevent errors*. … design the system such that users cannot make serious errors …. If a user makes an error, the interface should detect the error and offer simple, constructive, and specific instructions for recovery. …
6. *Permit easy reversal of actions*. … actions should be reversible. This … relieves anxiety ….
7. *Support internal locus of control*. Experienced operators strongly desire the sense that they are in charge of their interface and that the interface responds to their actions. …
8. *Reduce short-term memory load*. … displays [should] be kept simple, multiple-page displays be consolidated, window-motion frequency be reduced …

The rules of both Shneiderman and Molich and Nielsen can be traced back to a cognitive science background—models of human memory (comp. e.g.Anderson, 1994) and theories for categorizing human error (e.g. Norman, 1981; Reason, 1982; Rasmussen, 1986) provide an informative basis for possible design errors. In practice, these rules are often known to designers, but difficult to interpret in terms of a specific design context due to their high level of abstraction (Tetzlaff, & Schwartz, 1991). From these rules, only the last directly relates to reduction; it is, however, formulated in such an abstract manner that no direct guidelines for design can be deduced—functional, structural, and architectural minimalism would all be candidates to reach the goal of reducing short-term memory load.

4.3.2 Interface Guidelines and Minimalism

Interface designers refer more often to more concrete guidelines that provide reference and guidance in their daily work (comp. Souza, & Bevan, 1990). These exist in platform-

independent form (e.g. Smith & Mosier, 1986; Brown, 1988), but were more successful as platform-specific style guides (Computer, 1987; Microsoft, 1995; Inc, 1992). Although similar standards exist for other products e.g. the KDE environment (KDE, 2005), or the Qt library that the Gnome desktop is based upon (Benson, Elman, Nickell, & Robertson, 2004), the guidelines that are developed by Apple and Microsoft for their windowing systems MacOS and Windows have the longest history, and the largest user base in the industry (Apple, 1992; Apple, 2006a; Microsoft, 1999; Microsoft, 2006b).

In the *Apple Macintosh Interface Guidelines* from 1987, in addition to concrete advice about window design, interaction techniques, and even menu item placement, general design principles are provided in the first section of the book. „User control: The user, not the computer, initiates and controls all actions" is one of the first principles; its interpretation—asking whether the user really wants to invoke a function using alert boxes, however, is not as far-reaching as compositional minimalism would imply. The list of principles closes with „Simplicity" and „Clarity". Both are interpreted in terms of visual layout, e.g. for dialog boxes fewer choices should be presented, information should be made visible by both icons and text, etc. This interpretation is still current—the draft *User Experience Guidelines* for Windows Vista (ibid.) includes „Clean up the user interface: Remove clutter…" as one of the top 12 rules.

The call for visual clarity is clearly connected with reduction; it remains, however, on a superficial level of the interface—the guidelines do not mention application functionality. Interface design is here still ascribed a decorative role. Ample evidence, however, suggests that even the question of visual clutter is strongly connected to deeper design decisions. In Apple's most recent guidelines, the 80/20 rule (for a more thorough discussion, see Koch, 1999) is used to illustrate that trying to serve power users will often result in suboptimal interfaces. As the guidelines tell us, by trying to reach the other 80% of users, designs „typically favors simpler, more elegant approaches to problems" (Apple, 2006a, 29). Later, they even note: „The best approach to developing easy-to-use software is to keep the design as simple as possible" (ibid., 47). Simplicity is interpreted as reduction of visible features, and supplemented by „Availability: a corollary of simplicity" (ibid., 41).

4.3.3 Discussion

From a minimalist standpoint, neither the heuristics developed by usability experts nor the guidelines describing concrete system use relate to the standpoint defined by minimalism. Usability heuristics focus on cognitive limitations of the user, and intend to make designers aware of these. The thus limit their focus to individual aspects of the design, and although they suggest that mental effort on behalf of the user should be kept minimal, they fail to provide concrete advice how to do so. Guidelines, on the other hand, provide a wealth of examples and advice. Simplicity is here, however, understood in mostly visual terms, aiming at a „clean" interface. Although the importance of functional simplicity is also recognized by the Apple Guidelines, the notions of structural, architectural and compositional minimalism as alternative perspectives for creating simplicity even for functionally complex designs are not covered.

The role of superficial clutter and deeper design questions is further examined in the next section, demonstrating that the minimalist perspectives are echoed in existing HCI lore (4.4). Different conceptions of design are used to position the minimalist standpoint in the HCI discussion (4.5), and consistency, which is prominently featured as a guiding value in every set of interface guidelines, is then set in relation to minimalism to clarify its use as a principle guiding design (4.6).

4.4 Different Takes at Simplicity and Minimalism

> Persuading through Simplifying—Using computing technology to reduce complex behavior to simple tasks increases the benefit/cost ratio of the behavior and influences users to perform the behavior.
> — B.J. Fogg (2002)

The visual simplicity of interfaces is often directly connected to their ease of use. Bruce Tognazzini illustrated the need for simplicity drastically: „Designers learn in 6 months, users have to learn the same in 6 minutes. „ (Tognazzini, 1992). He emphasizes the importance of visual simplicity for design: „In the Macintosh, we try to be as sparing of words and extra visual clutter whenever possible, but sometimes we all overdo it. (Our original control panel … had no words explaining what the icons stood for, and everyone was confused. …)" (ibid.).

Jakob Nielsen also interpreted ‚less is more' in *Usability Engineering* in feature terms – "every single element in a user interface places some additional burden on the user in terms of having to consider whether to use that element. Having fewer options will often mean better usability". (Nielsen, 1993, 15) According to Nielsen (1994), „Aesthetic and minimalist design" means: „Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility".

Simplicity and reduction are commonly associated with „good design" and „high usability". The following sections collect evidence that interface design must be more than decoration, that access to functionality is determined by the visibility of interface elements, and that structure for access to information and functionality is often linked to simplicity.

4.4.1 Deep Design: Causes of Clutter & Excise

> Both the internal design and the user interface can have simplicity, but the interaction between the user and the program is what concerns us here, and frequently we have to sacrifice the simplicity of internal design to make the user interface simple.
> —Paul Heckel (1984)

In 1983, Donald Norman proposed a separation of interface design and „other programming tasks. Make the interface a separate data module … then the interface designer could modify the interface independently of the rest of the system. Similarly, many system changes would not require modification of the interface." (Norman, 1983, 2) Nowa-

days, we often have a division of labor, with some designers creating the interface, and software programmers devising the application logic. The positive result of this separation is the inclusion of designer skills—previously, design was considered an additional task for programmers. However, the fact that the underlying architecture influences the visible interface (german „Oberfläche") limits the effectiveness of a clear separation. Often, a design should influence the application logic, yet in real projects, the influence is often from the programmers and engineers who can determine design decisions.

Alan Cooper started his career as a programmer (Cooper, 1996), but turned into a critic of his old trade as he became interested in design. In *The Inmates Are Running The Asylum* (Cooper, 1999), he describes how an *implementation-centric* perspective contributes to unusable software through uninformed analysis of tasks and system-centered development processes: those who engineer applications should not be trusted; they tend to regard their own conceptual model as the only possible perspective. In *About Face 2.0* (Cooper, & Reimann, 2003), a practitioner's handbook for designing software applications, Cooper recollects his design experiences and presents a variety of design guidelines, iterating on goal-directed design (ibid., 12) and Personas, but also on his perspective of how software can be tailored to not impede on users' work.

The topic of reduction is addressed in a chapter called *Orchestration and Flow* (ibid., 123): for Cooper, the most important characteristic of „well orchestrated user interfaces" (ibid.) is that they enable the users' flow, a state of total concentration on an activity, losing awareness of peripheral problems and distractions (comp. Csikszentmihalyi, 1991). To create this flow, Cooper asks for less visible interfaces: „No matter how cool your interface is, less of it would be better." (Cooper, & Reimann, 2003, 119) In the tradition of Bødker (1990), he positions transparency as the prime value for interface design: „the interface must not call attention to itself …, but must instead, at every turn, be at the service of the user" (Cooper, & Reimann, 2003, 120). He further criticizes the existing ‚transparency' that exposes inflexible qualities of a system on the interface: „they show us, without any hint of shame, precisely how they are built. There is one button per function, one dialog per module of code, and the commands and processes precisely echo the internal data structures and algorithms" (ibid., 248). Instead of being able to work on their goals, users „*must* learn how the program works in order to successfully use the interface" (ibid.).

Cooper lists four guidelines that help reduce the visibility of the interface (ibid., 120ff), all working towards the provision of functionality without controlling work flow:

– *Follow mental models* involves the adaption of often-used functionality (e.g. menus) to the user groups' conceptual model; different users may have different perspectives on the same tasks and the same data.

– *Keep tools close at hand* complements this by pointing out that most applications are too complex to allow one mode of direct manipulation control all their features; Cooper advises to provide tool palettes and inspectors to allow direct access to functionality without the need for modes, and without forcing users to tidy up tools after use.

– In a similar vein, *modeless feedback* is asked for; status should be reported without interrupting the users' task flows.

– *Direct, don't discuss* finally stands for respecting the user's control; Cooper notes that while developers tend to view human-computer interaction as a two-way conversation, most users don't care to be interrogated by a program.

These guidelines are closely connected to what has been introduced here as structural minimalism. Cooper's key concern is „distinguishing possibility from probability" to prevent cluttered interfaces that make tasks more complex than they need to be. This touches upon the structure of interfaces—according to Cooper, not every possible function needs to be given equally accessible screen real estate, and hiding some functionality increases the accessibility of what is left (ibid., 124f). Examples he gives to illustrate the difference of possibility and probability include the „delete cells" dialog in Excel that always asks whether to delete all, formats, formulas or just notes—more than a minor disturbance (ibid., 124), or the Windows print dialog that does not separate printing from print setup, mixing configuration and invocation (ibid., 130).

Navigation is for Cooper something that should be avoided if possible. He goes as far as to suggest removing hierarchies in file systems (ibid., 156), and instead of folder inside folder, use grouping, search and attribute-based access (ibid., 204). Where structure cannot be avoided, it should be optimized: he demands designers should „inflect your interface" (ibid., 154). Under this label, he demands to minimize *typical* navigation. Yet his ideal system is not structurally minimal, „users make commensurate effort" for „fancy features" (ibid., 155)—on the other hand „users will be willing to tolerate … complexity only if the rewards are worth it" (ibid., 155). Cooper identifies three attributes that should govern the accessibility of features:

– *Frequency of use*: frequent items and tools should be no more than a click away

– *Degree of dislocation:* amount of sudden change caused by invocation of a command

– *Degree of exposure*: irreversible or dangerous functions need to be protected

Cooper's first attribute alone follows the ideal of structurally minimal access to functionality, while his other attributes introduce his understanding of protecting the user: both focus on minimizing possibly harmful or disorientating effects. While this is probably a good heuristic for a pleasant and safe interface, it might not be useful to overprotect powerful commands that create a large amount of sudden change.

According to Cooper, excise is harmful as it „stop[s] the flow" (ibid., 139ff). He gives the advice „avoid unnecessary reporting" and „don't use dialogs to report normalcy" (ibid., 129)—if the system repeatedly asks the obvious, automatization will prevent users from identifying critical situations, such as deleting the wrong file (ibid., 129). Cooper metaphorically calls this „the dialog that cried, »wolf!«"(ibid., 447f). All of these hints aim to reduce unnecessary access structure imposed upon the user by the programmer who unnecessarily requires entering information. Cooper finds the same words that de-

scribe minimalism to summarize his guidelines: „adding finesse: less is more" (ibid., 124). For him, all „navigation is excise" (ibid., 143), which includes windows, panes and menus mapping; scrolling should be minimized (ibid, 146), but also for different screens in applications should be reduced: „reduce the number of places to go" (ibid., 148). Cooper lists different types of excise: apart from the interruptions caused by the systems and navigation enforced on the user, he criticizes „training wheels" for „power users" (ibid., 137), „pure excise" (ibid., 137), e.g. hardware-management tasks that users should not have to deal with, and „visual excise" (ibid., 167) that he mostly finds in the Web in form of too many visual metaphors.

Cooper also gives some advice how to improve designs: Interruptions by dialog boxes could be minimized if ‚they' would „know where you are needed" and „Know if you are needed" (ibid., 412f)—dialogs that remember their last position and state, and do not appear if not necessary—unlike e.g. the „save changes" dialog in Microsoft Word that is raised by pagination after printing, not a user action, but a change by the program. Confirmations should only be used where a negative answer is expected. For all other cases, user actions must be reversible, and system exceptions should be handled by default procedures (e.g. when directory does not exist, or file is not writeable), although in some cases (e.g. overwriting a file), deep changes can be required to allow report and undo functionality (e.g. versioning file systems[9]) (Cooper, & Reimann, 2003, 449). Navigation could be reduced—although this would violate the „number of places" rule above—by breaking up functionality into smaller dialogs, not forcing the use of stacked tabs: „everybody hates it when the tabs move underneath the cursor" (ibid., 418f). Persistent navigation items like the tabs on the Amazon web site, and overviews such as the Adobe Photoshop navigator palette could further aid navigation. Users should be treated as „intelligent but with very little time" (ibid., 36) to avoid patronizing, and the problem of novice/expert interfaces is evaded as designers should „optimize for intermediates" (ibid., 35).

4.4.2 Visibility of Interface Elements

> Leibniz sought to make the form of a symbol reflect its content. „In signs," he wrote, „one sees an advantage for discovery that is greatest when they express the exact nature of a thing briefly and, as it were, picture it then, indeed, the labor of thought is wonderfully diminished."
> Frederick Kreiling, „Leibniz", Scientific American, May 1968

User interface design is more than decoration. Yet, observation of the visual attributes of an interface yields important clues about its functionality—the visibility of elements of the user interface determines the access to functionality. A balance has to be found be-

9. These are not to be confused with journaling file systems that keep a journal of I/O operations that can be executed after a delay or system failure, thus ensuring atomicity of operations and a consistent state. Versioning file systems were first introduced in the TENEX operating system (Bobrow, Burchfield, Murphy, & Tomlinson, 1972) to allow concurrent processes continue using old files; today, some experimental implementations exist (Cornell, Dinda, & Bustamante, 2004; Peterson & Burns, 2005)—and Apple announced TimeMachine, a journaling system, as part of MacOS X 10.5 (Apple, 2006c).

tween immediacy and richness of interaction. As Don Norman put it in his influential *Psychology of Everyday Things*[10], „[j]ust the right things have to be visible: to indicate what parts operate and how, to indicate how the user is to interact with the device. Visibility indicates the mapping between intended actions and actual operations." (Norman, 1989) He made popular the suggestion that design must anticipate possible actions and incorporate elements that make obvious both the functionality and correct use of a device.

The notion of „perceived" affordances , adapted from James J. Gibson (1979), lies at the heart of his turn towards design . In the tradition of cognitive science, trying to explain thought using symbolic processes, Norman suggests that attributes inherent in objects around us guide our interaction with them. Originally, Norman did not differentiate between Gibson's notion of affordance as an action possibility and the way that this possibility is conveyed —later he termed the notion „real affordance" for the former and „perceived affordance" for the latter. Although there have been suggestions to extend this typology further to include the sensory apparatus and differentiate between physical and functional possibilities for action , the differentiation between functionality and perception—as it was proposed by Gaver —seems to have had the most practical impact.

The perceived affordance of an object (or interface element) is affected by factors including its context in an environment or process, culture, or the influence of societal ‚norms' on the individual's understanding and use of an object, and the user's mental model, her understanding of and expectations for interaction with the object. Perceived affordances point out an important concept for designers—although not necessarily a revolutionary, as designers know that sometimes ‚form follows function' should be applied (comp. 4.1). Still, it is very useful as it relates a tool's functionality to properties communicating this functionality.

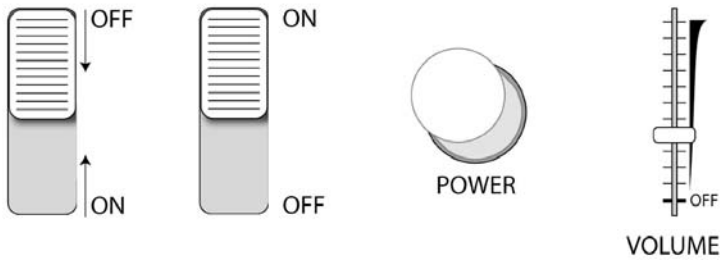


Figure 39: Power switches for stereos: mapping and affordance.

An example for conflicting affordances is illustrated in Figure 39: to switch off a stereo receiver, a switch has to be slided downwards (outer left); this is only obvious after both reading the text *and* interpreting the meaning of the arrows. A better mapping is created when the labels are switched, eliminating the need for arrows (left). A push switch (right)

10. This book, in a second edition more appropriately named *Design of Everyday Things*, has become part of the standard curriculum for interaction designers.

eliminates the need for label as its position communicates the state by convention. A volume slider that integrates power control completely eliminates the need for a power switch—lowering the volume eventually switches the device off (outer right).

Affordances point out where design must focus, they can be used to simplify the mapping of functionality and interface elements. Reduction plays an important role for designing for perceived affordances: In the given example, although the match with the user's mental model is probably quite fine, the arrows indicating the proper action are superfluous. Instead, the final states should rather be labeled as the nature of the switch already communicates the possible actions. The last solution is only best if the volume is changed often, it does not support switching the radio off and on to continue listening at the same volume. This is an example for a forcing function (Norman, 2002a, 133f). Physically combining two functions to force an activity can be problematic; norman gives the example of cars that would not start without seat-belts fastened: they mistook a parcel for a passenger, and most users disliked them so intensely that they had their mechanics remove them (ibid., 133f).

4.4.3 Access Structure

> A user expects that his programs be no more complicated to understand than the task they are doing for him
> —Paul Heckel (1984)

Functionality is not only exposed, or combined for forced access. On the contrary, it is often hidden in layers. These *access structures* become necessary when no direct mapping is possible, and interface elements must serve different functions that depend on the system's state. Different terms have been invented for designing access structure, ranging from the general interaction design to the more fashionable experience design. *Information architecture*[11] is a related discipline that aims to improve the users' access to information. An important aspect of all structuring approaches is the criteria they develop to differentiate between signal and noise, between the information or functionality a user needs, and what obscures relevant information or block access to functionality.

In *Designing Web Usability: The Practice of Simplicity* (Nielsen, 2000), Jakob Nielsen gathers advice on how to design Web pages, with special attention to screen real estate and transmission times. He summarizes his guidelines as „Simplicity should be the goal of page design. Users are rarely on a site to enjoy the design; instead, they prefer to focus on the content". (ibid., 97) According to Nielsen, visual design is thus an important element of information design. He propagates the ‚less is more' rule also for information content

11. The term might originate from Xerox PARC, which was founded in 1970 with the mission to create an „architecture of information" and has become popular with the rise of the World Wide Web when the structure of information became a problem for every Web designer (Wurman, & Bradford, 1996; Rosenfeld, & Morville, 1998).

of individual screens (Nielsen, 1993, 121)—Web site content should, according to Nielsen, be „succinct", scannable and partitioned into multiple pages[12](Nielsen, 1997; 2000, 101).

According to Nielsen, partitioning information is a good way to enforce this reduction. His guideline that text on Web pages should be split in smaller units connected by hyperlinks (ibid., 112ff) derives partly from his previous work on hypermedia (e.g. Nielsen 1995) , and partly from his emphasis on the visible part of Web pages in browser windows (see e.g. Nielsen 1995ibid., 18ff). He has continued to emphasize this guideline (Nielsen, 1999; Nielsen, & Tahir, 2002), and expanded it to also refer to forms that should be converted into „applications"—by which he seems to understand not the increased use of dynamic techniques on the Web, but a series on traditional non-interactive screens (Nielsen, 2005). The assumption that users do not scroll—that he bases his argumentation upon—could, however, not be reproduced in empirical studies; rather, it was shown that users do scroll if something interests them (Byrne, John, Wehrle, & Crow, 1999). And although a recent study found that clicks on links concentrate on the upper region of Web pages, an interdependence with the positioning of navigation elements is probable (Weinreich, Obendorf, Herder, & Mayer, 2006b). Yet, Nielsen's advice contains another aspect. If information is split „into coherent chunks that each focus on a certain topic" (Nielsen, 2000, 112), the user is enabled to „select those topics they care about and only download those pages." (ibid.) Thus, Nielsen's guideline can be understood as a call to adapt the information structure to the user's needs—minimizing the structure that permits users to access information, or functionality.

The ideal of structural minimalism as defined in this thesis marks out a specific approach to structure that does not necessarily yield a simpler system. However, what is—to the user—more important than the actual simplicity of the system is a perceived simplicity. Tognazzini noted „In my mind, perception is stronger than reality" (Tognazzini, 1992), when he was designing a small text editor: horizontal scrolling was perceived to be slow, and users complained. When the screen refresh algorithm was changed to first copy the visible parts and then display the new text—actually making display slower—an immense speed increase was perceived.

A structurally minimal system is thus not necessarily simpler, yet it may appear simpler to the user. John Maeda's *First Law of Simplicity* (Maeda, 2005c) illustrates an obvious mechanism: „A complex system of many functions can be simplified by carefully grouping related functions." Taken literally, Maeda's rule focuses on relations between functions, while structural minimalism suggests a focus on similarity between the *use* of functions. This difference can also be found looking at structuring techniques for menu systems (for an overview see Norman, 1991), e.g. top-down or bottom-up clustering techniques (ibid., ch. 11.3)(McDonald, Stone, Liebelt, & Karat, 1982; Chin, 1987), and the commonly used

12. Nielsen presumably based much of his advice on „writing for the Web" on a SunSoft usability study (Morkes, & Nielsen, 1998) where fictitious travel content was rewritten in different styles and judged by sample users.

novice/expert differentiation for application design that is essentially a primitive distinction between use situations. Information architecture draws on both users' static conceptions of categories—e.g. in the popular card sorting technique (.g. Fuccella 1997; e.g. Fuccella 1997Rosenfeld, & Morville, 1998; Hudson, 2005)—and the analysis of use situations, providing comprehensive support for navigation and offering search as an alternative, structure-less access method (Rosenfeld, & Morville, 2002).

This match between an interface and users' expectations has been described using the term „directness" (Hutchins, Hollan, & Norman, 1986). *Semantical directness* denotes the match of objects and operations between user and system on a conceptual level. *Articulatory directness* builds upon this and denotes the translation of user goals into interface commands. A further level of directness is described using the notion of *transparency* by Bødker (1990), who suggests that digital tools should let users work directly on their tasks, the interface becoming invisible.

Often, for different users different solutions are sought—as supposedly, their requirements differ. The common practice of designing two interfaces for interactive systems—one for beginners and one for expert users—is, however, not without criticism. Raskin (2000a) writes: „Designers speak in terms of »the tradeoffs between ease of learning and speed of execution in a system«(Card, 1983, 419). This may be true of particular interface designs, but I have seen no demonstration that it is necessarily a property of all interface designs … Present desktop GUIs are a compound of at least two distinct interfaces, a relatively visible and learnable but time-consuming menu-based system and an incomplete keyboard-based collection of hard-to-learn and unmemorable shortcuts. Two wrongs do not make a right." Raskin makes the point that users of complex systems cannot easily be rated on a scale between ‚beginner' and ‚expert', but rather focus on a part of the functionality, even to the degree of becoming an expert: „people may seek your advice on using it. Yet you may not know how to use or even know about the existence of certain other commands or even whole categories of commands in that same package. For example, a user of a photo-processing application who produces only online images may never need, or even learn of, that same application's facility for doing color separations, a feature needed primarily by commercial printers." (Raskin, 2000a) This analysis of system use clearly calls out for a modularization of functionality that is transparent to the user—architectural minimalism.

Instead of the two interface approach, Raskin proposes to create systems that are both *monotonous* and *modeless*. Monotonous systems have only one way to accomplish a task. In modeless interfaces, the mapping of commands and system reaction is static, a given user command has one and only one result. Raskin argues that „An interface that is completely modeless and monotonous has a one-to-one correspondence between cause (commands) and effect (actions). The more monotony an interface has for a given task space, the easier it is for the user to develop automaticity, which, after all, is fostered by not having to make decisions about what method to use." (ibid.). Modeless interfaces are an often sought feature: Alan Cooper remarks that errors can be prevented by using „rich modeless visual feedback" (Cooper, & Reimann, 2003, 451ff), giving the example of print-

ing in word processors; instead of raising a dialog box „some clipping will occur", the printable region could be highlighted within the document, immediately opening a path for user responses.

In practice, however, modeless interfaces are the exception—there are simply too many functions in most applications; not necessarily to the benefit of the user: although more functionality is often equalled with more power, a reduction of vocabulary can increase the power available to users. When David Curbow demonstrated the Xerox Star at ACM CHI 1998, he introduced the machine with „kudos to who was responsible for this—how many times have you seen a version 2 have fewer commands than version one?" (Curbow, 1998a), and praised it later at Xerox PARC as „from a user interface perspective, this interface [the Star] has still not been surpassed [...] in the *very few* commands that you need to do complicated things[13]" (Curbow, 1998b).

Direct manipulation (comp. Shneiderman, 1997; Hutchins, Hollan, & Norman, 1986) had two different effects: (1) the number of commands was reduced, increasing their individual visibility; (2) commands could be combined to allow sophisticated operations. GUIs can be understood as—on the level of interaction—architecturally minimal, and thus structurally minimal. They were superior to the command line because they „required a restriction on the range of vocabulary by which the user interacted with the system" (Cooper, & Reimann, 2003, 255).

4.4.4 Minimalism and Simplicity (again)

> Good design, like good writing, is simple and economical. Simple design makes the product easier to maintain and use. No magic formula will enable us to create simple designs; they are the result of creativity, false starts, and hard work. But when we can achieve most of our design objectives while making something simple, our results are likely to be good.
> —Paul Heckel (1984)

As the previous take at simplicity (3.2.1) demonstrated, the notion of simplicity is used by different authors to describe different qualities of interactive systems. Sometimes, it refers to visual qualities, although it is difficult to differentiate the superficial from the more fundamental design. Sometimes, it is used to describe the perceived affordances of non-physical interaction widgets, yet these seamlessly blend into access structures provided by the interface. A distinctly different take on minimalism is offered by Donald Norman in his book *The Invisible Computer*: „Simplicity design axiom: The complexity of the information appliance is that of the task, not the tool. The technology is invisible." (Norman, 1999a) An invisible interface certainly solves all problems of mapping and structure—or does an interface only become invisible after they have been solved? Bill Buxton adds: „My view is that very strong, specific devices are how you make computers disappear, and

13. The Xerox Star differentiated data icons and function icons, a function like copy could thus both mail and print documents, depending on the context.

how you change the computer from something that gets in the way into a prosthesis that helps empower us and actually makes the world simpler." (Anderson, 2000)

The question of appropriate functionality is deeply rooted in the design of interactive systems. Although not so long ago, computing was mainly thought of as computation, culminating in the famous prophecy ascribed to Thomas Watson, chairman of IBM in 1943, „I think there is a world market for maybe five computers", the PC revolution brought a powerful general-purpose computer to every workplace and home. The generality of the PC was heavily criticized as it required trade-offs impeding the usability for specific applications. Instead of general-purpose tools, computing machines were devised for specific tasks. Some attribute their invention to Vannevar Bush's concept of a *Memex device* that was conceived to access information, and build information paths (Bush, 1946; Müller-Prove, 2002). Jef Raskin, 1982 founder of Information Appliance Incorporation, is said to have developed the concept of *information appliance* (Raskin, 1982; Voelcker, 1986). The notion of information appliances was explained in more detail in *The Invisible Computer* by Don Norman (1999a), and has since entered HCI vocabulary. While the term was initially meant to describe a functionally specialized appliance that tried to perform a single function well, it is often used to denote more universal appliances—an example is the *universal information appliance* coined by IBM's TSpaces project (Eustice et al., 1999).

4.5 Minimalism and Conceptions of Design

The design of interactive systems is a multidisciplinary field. Consequently, manifold conceptions of the design activity accompany those who try to achieve usability: the field of Requirements Engineering follows an engineering approach, Human Factors tried to optimize the „human variable" in the organizational system, and user-centered design uses the technology-centered term „user" (Grudin, 1990b) to identify basic priorities. The latter perspective is illustrated by Bruce Tognazzini: „I was overprotecting my users. I kept them from straying from the safe, sure path I had plotted through my application. … In general, I drove the users stark, raving mad. Every time they tried to make a move, I was one step ahead of them, protecting them from any possible hurt. … Consider how you'd feel if every day when you came into work, Mom was there to make sure your nails were clean and your seat wasn't too high and you didn't have to much coffee. Mea culpa. This is what I did to my users." (Tognazzini, 1992)

Design has been described as an industry—as a form of mass production, a single application can be replicated and used by millions. It has also been described as a craft (Wroblewski, 1991, 7-12); according to Wroblewski, when software cannot easily be specified in formal terms „only craft-like efforts have succeeded" (ibid., 10). This does not imply that informed design decisions are inferior, only that „building human-computer interfaces involves applying the relevant knowledge in a complex problem solving context to tasks and artifacts too complex to be completely understood[14]" (ibid., 12).

14. T-shirts with the text „It depends", fashionable in the HCI community, and probably first introduced by

Yet, not only are many design efforts of immense complexity, but as design involves context, designers cannot even influence all parts relevant to a design. This is documented even by Jakob Nielsen—who is, judged by the number of generic recommendations and his acknowledgment of a preference for standardized guidelines ruling Web design (Nielsen, 2000, 217), unsuspicious of giving control away too freely: he observes a shift in the control over navigation that is created by the hypertextual origin of the Web (ibid., 214); designers cannot plan for, or control all aspects of navigation. Instead, users can enter a site via search engines on other pages than the home page, and bookmarks and backtracking may also allow them to navigate pages in a sequence that the designer never planned for.

Tognazzini changed his initial attitude to software design: „In my new designs, my users are truly in control. I assume they have an IQ that lies above room-temperature. … I assume they can make their own decisions on how they want to work" (Tognazzini, 1992). The computer assumes a role of a communications partner, and control is shifted to the user—commands are centered around the users' needs, instead of walking users through a „maze of commands" (ibid.).

The discussion of minimalism in art (chapter 3) identified a similar struggle for control, eventually leaving the artwork in control of context and user. Similarly, the role of designers changes to enablers of actions. This shift is echoed in e.g. the stage metaphor that Brenda Laurel (1993) introduced into HCI: A story offers a context; within the context, it offers activities and plots played by characters. In her transfer from her dramaturgical background to the design of interactive software, Laurel continues: „The users of such a system are like audience members who can march up onto the stage and become various characters, altering the action by what they say and do in their roles." (ibid., 16). The designers role is to set the stage, provide meaningful characters, and a draft plot. The storyline—what is happening on stage—lies within the control of users; control is not immediate, and not complete.

Also in 1993, Tognazzini (1993) drew a comparison of stage magic and usability: „Showmanship is the gentle seduction of the users, leading them to accept, believe in, and feel in control of the illusory world we have built for them." While Laurel concentrates on the interaction itself, Tognazzini also mentions the preceding step: a product must appeal to the user, or she will not buy it. The affective power of designing (perceived) freedom is also reflected in Don Norman's parallel between Boorstein's (1992) description of designing movies in *The Hollywood Eye: What Makes Movies Work* and his own three levels of design (Norman, 2004c, 112).

Jared Spool's User Interface Engineering company, represent a less scientific version of this statement.

4.6 Minimalism and Consistency

One of the strengths of the digital medium, it versatility, presents a major difficulty for designers. While physical tools need to honor physical rules, and the design activity in the physical domain is an intense interaction with limitations, digital designs enjoy far more freedom. David Gelernter rephrased this as „If you make a dam the wrong shape, it will crumble. If you shape a telephone like a Volkswagen or a tomato, it can still work divining the shape that seems inevitable, creating the »inevitability illusion«—the impression that you are looking at the pure visual embodiment of science of engineering—is an art pure and simple" (Gelernter, 1999). As Gelernter proclaims, „when the illusion succeeds, the outcome is technology that works beautifully and is beautiful" (ibid.). Without physical limitations, the design is bound by expectations—and consistency becomes an important aspect of design.

Consistency is probably the most commonly cited principle for the design of interactive systems. Almost every guideline lists consistency as an explicit value, and encourages consistency with ‚interface standards' using a multitude of suggestions, rules and examples. Almost every usability expert will agree that consistency is important, and heuristics for evaluating interfaces are bound to demand consistency. All the more, it is surprising that consistency is not well-defined. As Kellogg (Kellogg, 1989) points out, „Consistency has no meaning on its own; it is inherently a relational concept. Therefore to merely say that an interface is consistent or that consistency is a goal of user interface design is also meaningless". Even a committee of international usability experts was unable to agree on a definition after two intensive days of discussion (Nielsen, 1989). Grudin (1989) makes a case against consistency, arguing that without a reliable definition, the concept becomes only a secondary guideline. He argues that many successful interfaces are inconsistent and demonstrates that in some cases, consistency may even be harmful in interface design. His examples include keyboard mapping that demonstrate ease of use can conflict with ease of learning, menu behavior patterns that vary with different tasks (e.g. after copying text, paste should be selected, after italicizing a text, the italic attribute should be selected by default), and the form of consistency that exposes system architecture (e.g. the file system) to the user (cf. Bernard, Hammond, Morton, Long, & Clark, 1981; Grudin, 1988). Grudin previously pointed out that simplicity can be harmful if systems are simpler than the real world: „If the system does not preserve a distinction that users actually make, there will be no way for the Interface dialogue to match the user's expectations." (Grudin, 1986)

Despite his critical stance, Grudin tries to redefine consistency and differentiates between three types of consistency: *internal consistency* within an application, *external consistency* with other applications, and *consistency with familiar features* from the real world (Grudin, 1989). An ambitious formal approach of defining consistency was taken by Wiecha et al. (1989), Reisner (1990) or Rieman et al. (1994), starting a heated debate (comp. Grudin, 1992; Wiecha, 1992) yet as Chimera and Shneiderman (Chimera, & Shneiderman, 1993) note, the formal approaches„get more elusive as they are scrutinized".

In practice, informal factors were shown to have great influence. Tognazzini reported that the immediate availability of key applications, MacWrite and MacPaint in the case of t he Apple Macintosh, was vital for achieving interface consistency (Tognazzini, 1989). In *Tog on Interface* (Tognazzini, 1992), he advises a selective approach to consistency: „(1) If it ain't broke real bad, don't fix it. (2) Reflect the illusion of the interface, not the realities of the hardware. (3) Build on existing visual/behavioral language. (4) Use big concepts. (5) Depend on precedent. (6) Invent new objects, with new appearances, for new user behaviors".

Consistency is often connected with the use of metaphors, which is held to be an important factor for the initial appeal and learnability of a system. The desktop metaphor of the Xerox Star is a famous example: „Even on first contact, the machine will appear to be as familiar, friendly and easy to cope with as, well, as the top of the desk" (Xerox, 1983, 3:43). Consistency with metaphors, however, often does introduce unnecessary restrictions. An example is a calendar „month view" that prevents users from inspecting entries that cross the border of two months (Cooper, & Reimann, 2003, 30). Nielsen joins those who are critical of employing one, over-arching metaphor in design (Nielsen, 2000, 180): „the greatest weakness of metaphors is that they … push the site in directions that seem fun and appropriate within the metaphor but leave users' real goals behind." Most successful metaphors (e.g. the shopping cart for e-commerce sites) are, according to Nielsen (ibid., 188), successful not because of their inherited meaning, but due to the standardization that creates a consistent interface to shared functionality over different sites.

Cooper suggests to replace metaphors with idioms (Cooper, & Reimann, 2003, 248 & 250): „metaphors are hard to find and they constrict our thinking" (ibid., 252) Nelson called this „the design of principles" (Nelson, 1990)—learning idioms is compared by Cooper to learning the meaning of figures of speech. Cooper points out that often, we do not use interface elements because we immediately understand their meaning through the embedding metaphor, but because we learn how to use them by observing others: „Windows, title bars, close boxes, … hyperlinks … are things we learn idiomatically rather than intuit metaphorically" (Cooper, & Reimann, 2003, 250). Examples include interaction techniques, e.g. the computer mouse: „We don't know how mice work, and yet even small children can operate them just fine" (ibid., 251). He illustrates the extensibility of idioms with mouse clicking: a single click means placement, a single click combined with dragging characterwise selection. This functionality is intuitively extended by double clicking that selects a word, and double clicking and dragging that activates wordwise selection. Triple clicking, finally, invokes selection for paragraphs.

Tognazzini reported that on the Macintosh, a switch of metaphor from Finder to Switcher and back to MultiFinder (known as simply »Finder« again under System 7) was made. Switcher introduced a metaphor that „existing users had to form new and somewhat confusing mental models to be able to understand" (Tognazzini, 1992). In retrospect, the original metaphor was restored with MultiFinder: „It was far more powerful than Switcher, and yet, as a model, didn't require the user to learn anything new." For Tognazzini , this demonstrates „the importance of really considering the impact that fea-

tures added to the second or third release of a product will have on the metaphor" (ibid.). Here, the *stability* of the work environment is the cause for consistency. This is further illustrated by the Dennis Austin, the creator of Powerpoint, originally an application for the Macintosh: „The Apple Desktop Interface … defines a number of consistent graphic elements (menu bar, window border, and so on) to maintain the illusion of stability. …" Hiding the menu bar resulted in severe disorientation: „I have seen, during user testing, the very real fear etched on the face of users when the menu bar disappears. I have watched them literally panic as they are trapped in a strange world" (quoted in ibid., 34).

Consistency can serve as a replacement for simplicity. If most of a new design is already known, or consistent with a known design, only new additions have to be learned and mastered. Thus, the impression of even extremely complex systems can be simple—if they adhere to standards set by legacy systems. The disadvantage of consistency here is clear: it inhibits change, and better solutions have a hard time to become accepted—or are never adopted because it seems more complex to relearn conventions. One well-known example is the Dvorak keyboard, which is often considered to be superior to current QWERTY-type keyboards, but never even had the slightest chance of commercial success (Diamond, 1997; Liebowitz & Margolis, 1990; Brooks, 1999).

On the World Wide Web, consistency has been an important argument for the increasing use of Web-based applications. To some degree, Web browsers unify the use experience for these application. However, their interface is still based on the Hypertext document metaphor and lacks support for the the requirements of interactive applications (Weinreich, Obendorf, Herder, & Mayer, 2006b; Weinreich, Obendorf, Herder, & Mayer, to appear). This illustrates that consistency alone is not always sufficient, but that consistency with conventions limiting necessary interaction can be very harmful for usability.

To summarize, consistency remains for the design of interactive systems an important concept of relative nature. According to common agreement, „users' tasks and application domain are a major focus for providing consistency" (Chimera, & Shneiderman, 1993). This very generic claim can be specified in the light of architectural minimalism by demanding that individual tools need to form a consistent whole. For individual tools, and in some cases—as the concept of structural minimalism indicates—other factors, such as immediacy, can become more important for a useable design than consistency. Although metaphors can be helpful, they are also a dangerous tool for design as their analogical function places restrictions on digital design that can limit usability. The stability of an interface is an important motivation for consistency, and thus finding initial metaphors becomes a crucial part of design; this is even more critical for systems with minimal functionality—if later functionality is to be expected, the chosen application model needs to allow for an expansion.

4.7 Revisiting the four notions of Minimalism

This chapter illustrated some of the existing perspectives on design that have long taught and practiced reduction. The discussion of industrial design (4.1) demonstrated that the practice of interaction design draws many of its methods and concepts from the practice of industrial design, and that the differentiation of function and decoration was always difficult; although this discussion does not explicitly focus on aesthetics and joy of use, these are influenced by functional design.

International standards were cited to illustrate the difference of the minimalist perspective. To enable constructive design, there is a need for explaining the objective measures defined for the ease of use in standards. While use qualities define aspects that are desirable in designs, the minimalist standpoint introduces a shift in focus to a more constructive stance that examines not only aspects of a single system, but a broader use context (4.2). An examination of the often-cited ‚Golden' and ‚Silver' rules of Shneiderman and Molich and interface guidelines for different platforms further illustrated the limitation of the current focus on functional and structural aspects of a design, and the interpretation of simplicity in purely visual terms (4.3).

The discussion of aspects of the heterogeneous literature on HCI that touch upon reduction indicated that minimalism describes aspects of simplicity, which exist in the literature, but are understood in inconsistent terms: *orchestrated flow*, *affordances*, and *information appliances* are all shows to relate to different notions of simplicity (4.4). To anchor the minimal standpoint in the existing literature, different conceptions of design are discussed, arguing for understanding design as a *craft* (4.5), and the notion of c*onsistency* was discussed as a possible replacement for simplicity: while it reduces the amount of work needed to learn the use of a new tool, possibly harmful effects of consistency are illustrated as it inhibits the evolution of design (4.6).

The conflict of complexity and simplicity is identified as a recurring theme, experts agree that a reduction is often necessary, yet the degree and direction remain debatable. The four notions of minimalism put forward in this thesis help to identify areas of reduction. Revisiting them individually, it becomes clear that *functional minimalism* is often sought for, and closely linked with *information appliances*, a concept of simple tools that perform only a single function[15]. *Structural minimalism* is one of the most popular interpretations of simplicity—transparency and directness are positive values for design. These values, just as ‚suitability for the task' defined in ISO 9241, however, give no constructive advice and can only be evaluated indirectly through the user—if it fits, it fits. *Architectural minimalism* and *compositional minimalism* present novel concepts—although the necessity is illustrated by the discussion on access structures and modelessness (4.4.3), and the values defined in in international standards (4.2).

15. It is, however, doubtful that pure information appliances do really exist.

Simplicity, and reduction, are often mentioned in guidelines and heuristics, and expert advice often relates to options for reducing complexity. This illustrates the need for an understanding of reduction and reductive methods; there is, however, no systematic examination of reduction in the literature: only specific aspects, such as the visibility of interface elements, the problem of modes[16], or the mapping of interface elements and functionality have been formalized in terminology and descriptive theory. Minimalism tries to fills this void as it sets out to differentiate different notions of simplicity, and presents an alternative to the exclusive use of techniques building on consistency and metaphor.

Taken together, the different notions of minimalism form a standpoint for design. This standpoint is marked by a shift towards a both holistic and fragmenting perspective—designs need to reduce their individual complexity to allow designing complexity on a second level. The scope of HCI analysis widened as the object of research includes not only a single application and its users and context, but networks of interoperable tools. A second shift induced by the minimalist perspectives leads away from interface design, and towards system design. This demands a conception of interface design as a deep-reaching discipline—away from decoration towards functional analysis.

16. Modality can be considered a method for attaining structural minimalism as it decreases the number of options for the user. It does, however, also increase the amount of composition as the designer plans in advance what actions users are allowed to perform—this is a central aspect of critique for modal systems.

5 Minimalism in Products

Designing products for the mass market is inherently difficult. Some designs work better than others, and while public apprehension of design is certainly subject to fashion, mode, style and vogue, *minimalist* design has regularly been blessed with some success; in recent history, it even seems to have become fashionable: Philips advertises „simple is beautiful", mobile communications companies introduce no-frills tariffs using the „less is more" motto and, last but not least, the profits Apple Computer wrings out of the „stylishly minimal" iPod music player series seem to have no end.

Different designs claim simplicity for themselves, while the number of interpretations of the term approaches the number of design examples. In this chapter, a selection of existing designs was made to illustrate that minimalism delivers a terminology that helps to analyze designs, and to further clarify the meaning of the individual notions of minimalism. The selected examples come from a wide range of designs that were attributed with the label ‚simple' by the public; they were chosen as they exemplify at least one aspect of minimalism, and while there are aspects of all notions of minimalism in almost all designs, this discussion tries to highlight the most prominent aspects of a design.

The discussion in each section starts with an „artificial example" that was selected to illustrate the direction of reduction. Following it, several examples are discussed whose main design tendency is identified as belonging to the notion of minimalism in discussion; in these examples, other notions of minimalism are often discussed, as well. Each design discussion is assessed in minimal terms—without intending to rate design quality. Instead, minimal qualities are exposed, and minimal terminology is sharpened. The closing example is always chosen from the domain of word-processors to demonstrate the applicability of all minimalist perspectives to a single application domain.

5.1 Functional Minimalism

As functional minimalism is perhaps the best documented form of minimalism, it will be used to start discussion in this chapter. But, before the practice of Apple Computers to acquire owners of complex software suited and release functionally reduced versions, the

CommSy CSCW system, and a first aspect of word processing are discussed, the idea of functional minimalism shall be reassessed by looking at analog tools: the knives of the traveller and of the cook.

5.1.1 Cutting Edges

The archetype of the multi-functional tool is a knife that was never used in the manner it is named for: the „Schweizer Offiziersmesser", or more aptly, the swiss army knife—the Swiss army uses a much simpler and sturdier metal-plated version with only 4 functions; officers never received a dedicated knife (Jackson, 1999). The knife's design approach appears to be to integrate every possible functionality into the pocket knife—which will always be with you, anyway. Swiss army knives (Figure 40) are able to open cans, turn different types of screws, uncork wine bottles, saw wooden logs, do needlework, cut and file fingernails—the list is almost endless, as specific knives were produced for many target groups, and the most powerful knife integrates 33 different functions (the XAVT special model even boasts 80 functions). After September 11th, 2001, and the consequential intensification of airport security protocols, sales dropped sharply as sharp knives were banned from hand luggage. The company's solution was consequently following previous policy: a USB drive pocket knife was produced that no longer had an edge.

Figure 40: Swiss army knives: the most powerful serial model, and the special XAVT model.

On the other side of the spectrum, one finds the multitude of knives a professional cook will use. As an example, some knives commonly used to prepare sushi are depicted in Figure 41: while there is an „almighty", all-purpose knife (3rd from right, Kurouchi Santoku), special knives are built for very specific tasks: from left to right, the Sashimi knife is for slicing raw fish, the Nakkiri knife for cutting vegetables, the Kurouchi Mioroshi Deba for cutting fish with bone, the Korouchi Santoku for cutting everything else, the Mukimono for paring vegetable and fruit, and the Kaiwari for extracting meat from shellfish. Blades vary in form, length, thickness, and steel and are built to specifically cater to the needs their appointed task requires.

Figure 41: Assortment of Sushi knives.

These knives are all specifically built for a single purpose; their focus on very few tasks allows to specify exact requirements and construction can be tailored to meet these specific needs. Although each knife only fulfills a single function, it will do so much better—and to greater satisfaction of the user—than a multi-purpose knife[17].

Discussion

Functional minimalism first and foremost means a reduction of functionality—not necessarily a common goal of information science. Tools for professionals—many from the physical world—the woodworker's shop, the many instruments of a percussionist, or the utilities of a system administrator, and here specifically the knife example demonstrate that *functional minimalism* can be translated in terms of a quality trade-off: a reduction of the number of functions allows an *increase in the quality* with which functions are performed. Reversely, adding additional functions to a tool will *compromise* existing functionality. The example also demonstrates the close relationship of functional minimalism and architectural minimalism—if functionality is distributed among tools, a reduction of the functionality of individual tools is the logical result; and reciprocally, reducing the functionality of tools creates the need for a concerted ensemble of complementing tools.

17. The reason for the recent appearance of many interaction designs that focus on a specific functionality is related to the popularity of the term information appliance—although in its absoluteness, the term never fits; most devices tend to accumulate functionality—a PDA acquires telephone qualities, mobile phones are used to access shared calendars. In the literature, this tendency is described as *convergence*.

5.1.2 Apple GarageBand (i-Series 1)

Apple is often entitled not a software company, but a design company. Traditionally, it provided both hardware and operating systems, but the design of productive software was—but for some notable exceptions, such as Hypercard—left to third partied (e.g. Microsoft for the office suite). However, with the introduction of MacOS X, Apple began packaging software with the operating system: first, small appliance-like tools such as an email client, an address book, and a Web browser were included free of charge. Subsequently, bundles of multimedia (iLife) and office (iWork) software were being sold by Apple. Some of these products were created from scratch by small development teams—e.g. iMovie 1.0 was finished with a team of three programmers (Sellers, 2005)—but most were not built in-house at Apple, but resulted from buy-ins of smaller companies; e.g. technology for iTunes—in a product called Soundjam—was acquired from Casady & Greene in 2000 (Smith, 2005), and GarageBand was developed on the basis of Emagic technology(Orlowski, 2004; Boddie, 2005)—before the acquisition by Apple, Emagic was the second largest player in the market of electronic music composing software (Inc., 2002).

Description

GarageBand demonstrates both the marketing and the design approach taken by Apple. As Apple previously used a three segment approach to the video market, installing iMovie/iDVD as consumer applications, Final Cut Pro that was acquired from Macromedia as mid-range products, leaving competitors such as Adobe only the High-end market, the music market is now also served by an entry-level product, GarageBand, an intermediate Logic Express, and the professional Logic Pro (Leishman, 2004).

What is most interesting in terms of design is the use Apple made of the sophisticated music recording technology that was acquired from Emagic. Instead of continuing along the lines set by competitors, e.g. with Cubase SL3 that was advertised with „50 new features“ (Steinberg, 2006a), and providing what is essentially a crippled version of the professional product able to record fewer tracks, use less voices and exclude certain high-end features (Steinberg, 2006b), Apple chose to follow a much more rigid path. The entire application was redesigned to match other i-Series products in look-and-feel, and the features included in GarageBand were reduced to include only the absolutely necessary—in this context, a quote attributed to Apple CEO Steve Jobs relating to iTunes might explain the direction of design: „…we don't want a thousand features. That would be ugly. Innovation is not about saying yes to everything. It's about saying NO to all but the most crucial features.“ (Steve Jobs quoted in Sievers, 2004) Features dropped for GarageBand included a score editor, MIDI import and output, most sophisticated MIDI manipulation functions, tempo or key change within a song, and strictly limited plugin support (Macjams.com, 2004b). What was left was an audio sequencer that could be used in combination with almost unprocessed MIDI tracks.

Apple made every effort to conceal that more powerful features could exist while it was very much in the interest of its competitors, e.g. Steinberg, to illustrate that there existed a better, albeit more expensive version of their software that users could upgrade to. As a

consequence, the „breaks“ in the design are not visible, and the software was design as a self-sufficient „whole“. The interface of GarageBand (Figure 42) is relatively simple—at least if compared with other music recording software (comp. Figure 43): there are fewer overlapping controls, and the number of different screens is significantly reduced; most of the work in GarageBand is done directly in the main window. Integrated instruments offer less options for tweaking the sound, but rely on presets useful for most home recording needs. All in all, most of the complexity that burdens other recording software was stripped off, leaving only those features that are most necessary for GarageBands key functionality, that of quickly arranging lead and arrangement tracks.

Figure 42: Apple Garageband 1.0 screenshot.

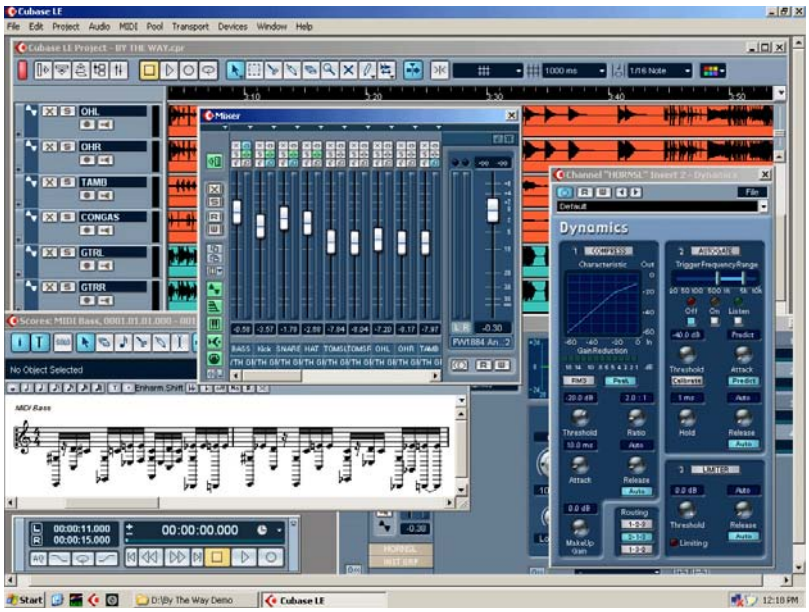


Figure 43: Cubase LE 3.0.

At the same time, the concept of audio loops, segments of recorded audio that could be combined to form patterns, with a huge library of pre-produced patterns designed to form an accompaniment to self-conceived melodies, was introduced. Thus, even without all the features dropped, hobby musicians were able to quickly generate professional-sounding tracks, and community Web sites blossomed to facilitate the exchange of music (Macminute.com, 2004)[18]. With the loop concept, Apple introduced direct-manipulation into the field of music production as patterns can easily be moved (arranged), resized (time-streched, or looped), dragged from the repository, or copied from other projects.

The result for the user is a minimal amount of distraction, enthusiastic reviews of GarageBand demonstrate that even professional musicians use GarageBand for sketching new song concepts „with a minimum of hassle" (Macjams.com, 2004a; see alsoPogue, 2004; Preve, 2004; Schumacher-Rasmussen, 2004). Referring to Csikszentmihalyi's (1991) notion of flow, musician Spencer Critchley states: „Why not just start in Logic, since Logic can do everything Garageband does, plus about a universe-full of other things? Because in flow, every second counts." (Critchley, 2005)

Due to its simple functionality and the conscious limitation of features, GarageBand seems to be an attractive tool for sketching musical concepts. And as some things, such as adding many layers of self-recorded sounds upon one another, become very simple through the focus of the software, users report „it also inspires me to do more. To experiment with layering. To arrange new songs. To create things that I wouldn't otherwise have the resources to create" – GarageBand is perceived as „so easy to use and, in several important ways, powerful as well" (Nagel, 2005). This combination of simplicity and power, Gelernter's *machine beauty*, allows both novices and experts to very successfully use GarageBand as a tool—always within its narrowly defined scope of competence (Boddie, 2005).

On the other hand, GarageBand uses the same technology internally that Logic uses, making it seem natural for ambitious musicians to continue work within the same product family; thus, providing a low-cost alternative with minimal functionality might be generating more profit for Apple as two products are bought by customers and used in different stages of production.

Discussion

The Apple i-Series move the focus on the speculative and the temporal nature of minimality, two important aspects of the notion of minimal functionality: the minimal set of functionality is not only difficult to determine, and cannot simply be found by reduction alone, but it is also subject to changing demands and requirements of the users.

The GarageBand case highlights the question how minimal functionality can be defined; as opposed to the Steinberg line of music sequencers, minimal functionality will not be

18. Among the most active GarageBand communities are iCompositions (http://www.icompositions.com/) and MacIdol (http://www.macidol.com).

reached by placing artificial restrictions on the underlying engine while keeping the same interface (comp. Figure 43). Instead, Apple succeeded with GarageBand because it designed a new application—that builds upon a common engine, but exposes only selected features. The success of this strategy depends upon the ability of the designers to predict which selection of features forms a ‚core' that is complete in its simplicity.

Yet, GarageBand also highlights that later product versions tend to accumulate more features even for a functionally minimal application: In GarageBand 2.0, new features were introduced. Some of these follow from the paradigm shift that GarageBand introduced with its emphasis on loops: New loops can be created and the categorization of existing loops can be altered by the user; this makes the new paradigm more useful, and although it adds some interface complexity, it provides a simple opportunity for users to tailor the existing interface and adopt its structure to their tasks. Some features are useful without complicating the interface, e.g. recording up to 8 audio channels simultaneously with appropriate hardware. Other features, however, undo simplifications of the interface: e.g. GarageBand 2.0 (re-)introduced a score editor, thus restoring a feature common to competitive products. The duplication of functionality in the piano roll and score editors makes GarageBand more complex—yet, it was often remarked that without it, it would be no real match for musicians (e.g. Morgan, 2005)—although other applications exist that provide exclusively functionality for arranging and printing scores, something even the newest GarageBand does not do very well.

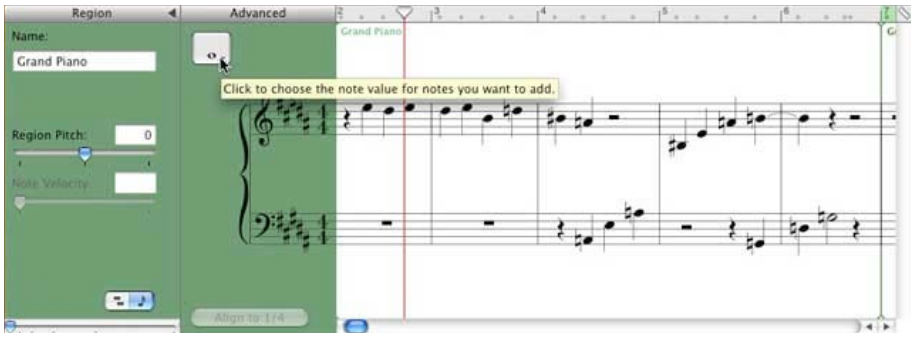


Figure 44: Score editor added to GarageBand 2.0.

Other i-Applications demonstrate a similar procedure, e.g. iPhoto 2.0 boasts a „host of new features (Pogue, & Story, 2006), and Keynote 3.0 introduced 3D-charts, new transitions, additional views, and many more (Tessler, 2006). This *2.0 syndrome*—the differentiation by additional features—can become dangerous for an application suite that tries to deliver the essential functionality with a clean and simple interface.

Apart from the risk of ‚adding more of the same', another risk lies within the changing focus of a tool. While a craftsman's tool is reserved for a single, unchanging purpose, digital tools are often required to also support the latest fashion. GarageBand 3.0 again added functionality, but this time with a new focus: as *podcasts*, audio files distributed using RSS or Atom feeds and stored on computers or mobile players, such as the iPod, were

becoming increasingly popular, Apple chose to add some features to GarageBand that would ease the production and publication of podcasts (Breen, 2006)

Table 6: Minimal Aspects highlighted in Apple's Garageband

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: Relative to its competitors, GarageBand provides a drastically reduced functionality. Its development history demonstrates that shrinking the feature list of a product can create a tool that is both successful in the mass market, and also used for specific purposes by professional experts.

Structural Minimalism: The interface of GarageBand is very similar to that of other digital music recording software—e.g. it also simulates the physicality of real studio equipment. Its simpler structure is mainly effected by the reduction in functionality. GarageBand also provides fewer „views" on the data, the missing score editor demonstrates the difficulty of differentiating between structure and functionality.

Architectural Minimalism: In the professional use of GarageBand as a specialized tool for musical prototyping, GarageBand can be seen as part of the Apple music suite. It does not, however provide a clearly distinct functionality from the more sophisticated applications, as innovations in GarageBand were later incorporated in the more expensive members of the series. The differentiation between tools is here created by GarageBand's focus on speed and semi-automatic accompaniment, with the Loops concept introducing a rapid technique of composing music to mass market sequencers.

Compositional Minimalism: GarageBand's field of competency is clearly defined as music production, and users generally use it only for this purpose. GarageBand 3.0, however, enabled many amateurs to produce Podcasts—streamed recordings of voice and music—and contributed to their steeply rising number; Podcasts are used as a generic type of media to record university courses, create autobiographical histories, blog on audio, or distribute personal music compilations, much like the mix tapes of years ago.

5.1.3 The CommSy Community System

CommSy is a web-based groupware designed to support communication and coordination in working and learning groups with a focus on the exchange of documents and the sharing of important notes and appointments between users. Comparable systems include e.g. BSCW (Bentley, Horstmann, & Trevor, 1997), phpBB (phpBB, 2006), and Moodle (Hilgenstock, & Jirmann, 2006); yet, in contrast to other CSCW systems, simplicity was defined early in the development process as a primary goal. As a consequence, the CommSy development process explicitly aims to question the usefulness of new features (see 6.3), and although the first version was presented in May 1999 and CommSy can be considered a mature software product, it lacks many features other systems provide: there is e.g. no sophisticated access control mechanism, and ownership rules are very

simple (Jackewitz, Janneck, & Strauss, 2004; Obendorf, 2003); closed groups are supported through ‚rooms' with restricted access; within a room, an information entry is either writable exclusively by its creator or writable for all while everything is readable for all. CommSy also refrains from imitating hierarchical folders common in „desktop" interfaces; due to technical restrictions—drag and drop does not work as expected, exchange with the desktop computer is cumbersome—a folder metaphor can currently not be fully implemented by a Web application. Furthermore, folders define a single fixed information structure that may present problems to different users.

Description

CommSy is trying hard to minimize functionality, following the ideal of *functional minimalism*. Its primary function was storing documents for closed groups, and offering equal access to these. Quickly, other types of information items were integrated: appointments, news announcements, persons, and groups. Although it is now used outside university courses, and other contexts have created new requirements—the increased use e.g. in secondary schools led to changes in wording, and users from virtual networks and commercial companies demanded the integration of more advanced functionality, such as graphical calendars and active assignment of users to tasks—the initial concept, storing information items in time-ordered lists, has scaled surprisingly well for new users from other contexts.

Feature requests reaching the development team were often discussed by the team as a whole, and when no majority could be found in favor of the request, it was put off until the requesting stakeholders either found a way of appropriating CommSy to their needs, or enough (financial) resources were found that the specific feature would be implemented. Features were even removed from the system when too few members of the development team found them useful, or too little use was observed; they were either integrated with other, existing functionality, or made optional. An example for the former was the ability of creating structured documents; initially, this was introduced as an additional category, but when little use was observed and the team found users did not understand the concept, the functionality was integrated with the existing document type of information items. An example for the latter is the already mentioned calendar: because it was vehemently required by some users who were willing to pay for development, it was implemented and integrated with the CommSy system. But as it seemed to make little use for most contexts, it is hidden by default. Other parts of CommSy functionality can be configured by users: e.g. the extent of displaying news items can be configured to show either individual news titles, just the news category, or nothing. Configuration, however, is limited to the core functionality of CommSy as even the typical room coordinators have little or no background in computer science, and little tolerance for unnecessary complexity.

Structural minimalism in CommSy is only visible on second glance. It is, however, one of the main strengths of the system and has grown through year-long refinement of individual categories. Screens for different categories display slightly different selections of information, and the placement of information is optimized to allow easy reading and intuitive understanding. This became apparent when the system, that was initially

programmed in PHP5, was migrated to Java-based technology (Jeenicke, 2005). Within a single semester, the functionality for a single category was implemented almost fully, and an integration with the PHP-based system was possible (ibid.). It was, however, far more difficult than anticipated, to „get the design right" (Finck, personal communication): the automated test suites that were used to guarantee correctness of the re-implementation failed to detect subtle details of placement and differences between categories. This made it necessary to return to manual testing, and also contributed to an unexpected prolongation of the migration process.

It is a matter of perspective whether to attest architectural minimalism to CommSy: the underlying PHP code is structured in a manner that every requested view is catered for by a dedicated ‚page' module; these ‚pages' combine views on information item to form the designed layout and these in turn refer to ‚data' classes that model the underlying structures. Thus, CommSy makes no attempt to introduce a global partitioning of functionality, and additional functionality is not integrated using a plugin mechanism, but by insertion of control structures in the affected ‚page' modules.

On the other hand, the ‚pages' served by CommSy are structured into categories that are presented to the user (Figure 45). There is some evidence that different categories are recognized by the users as different tools for different uses. Typical questions to the user support include the question what the different categories should be used for, and how to best combine their use; this information was also made available in an introductory guide (CommSy, 2005). This argument is further supported by anecdotal evidence: the introduction of a ‚document' category that allowed users to add sections to existing documents failed as its functionality was not considered to be different enough from the existing ‚material' category; the new „tool" was never accepted, and consequently merged with the existing category.

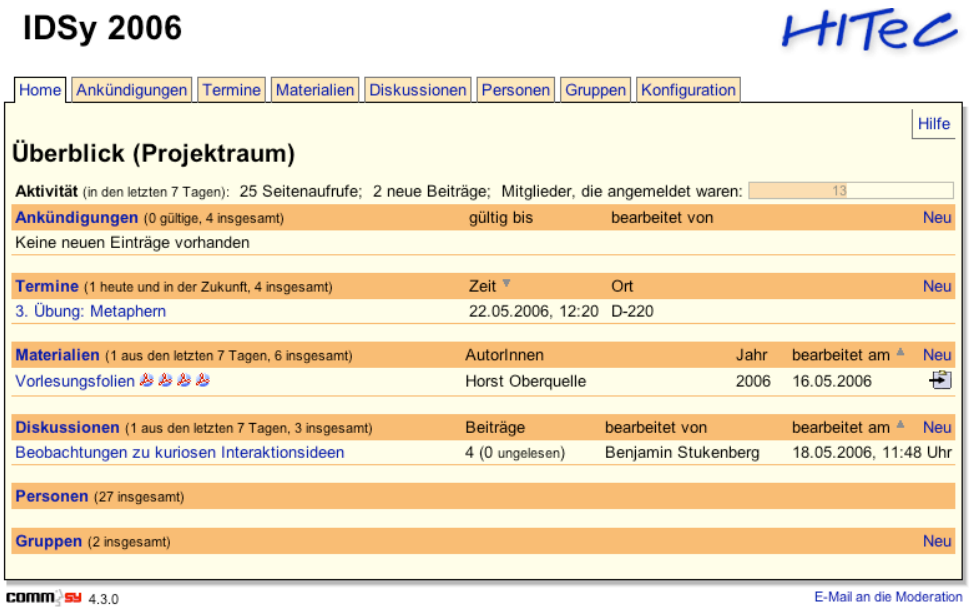


Figure 45: CommSy main page with access to categories via both links and tabs.

The use of categories alone does also not qualify CommSy for minimalist structure. Rather, the aforementioned year-long process of incremental refinements has stringently created an access structure that is now adopted so well to the users' tasks that it qualifies for structural minimalism. Individual ‚pages' optimize placing and filtering of available information—a good example is the ‚network navigation' introduced with version 3.0 in 2004 (Janneck, Obendorf, Finck, & Janneck, 2006): while previously, access to information items was only possible via lists of items, the ‚network navigation' introduced direct associations between different items (Figure 46); however, as developers and users felt that only some associations were useful, they selection and order of association types differs with the type of information item displayed (Obendorf, Finck, & Janneck, in prep.; Janneck, Obendorf, Finck, & Janneck, 2006).

Figure 46: CommSy group detail page with ‚network navigation', presenting links to associated items.

This optimization aims at minimizing interaction complexity; as unnecessary choices are eliminated, functionality is reduced to a ‚core'. CommSy is an interesting example for the structural similarity of dealing with similar information items in different contexts as new users quickly identify this grown ease-of-use as one of the software's key qualities, and have only few problems arising from differences in workflows—often, their critique focuses on functionality that they feel is necessary, but that is unavailable within the software.

Part of this extended functionality, according to an initial assumption of CommSy development (use within a media mix, see e.g. Pape, Bleek, Jackewitz, & Janneck, 2002) would be provided by other tools. Its developers regarded CommSy as only one tool in the portfolio of university teachers and students; depending on the type of course, other tools would supplement, or even replace the functionality of CommSy. One example for this is the limited email functionality within CommSy. As users already employ a dedicated application for sending and receiving email, replicating part of this functionality within the web-based system would create problems of redundancy and ambiguity: it can no longer easily be determined where an email was sent, or where one was received. While there is some evidence that some users move important data from their email to dedicated places, such as a calendar (Gwizdka, 2004), there is a general consensus that email is a unifying application (Whittaker, Bellotti, & Gwizdka, 2006; Bergman, Beyth-Marom, & Nachmias, 2003; Boardman, & Sasse, 2004). Consequently, introducing an additional application to manage email seemed unprofitable; making the single existing application responsible was preferred to avoid making organization of received and sent email more difficult for users.

In contrast to other systems, syndication was also opposed on the same terms—a motivating effect for users to keep themselves updated was assumed: for competitors, such as BSCW, extensions have been developed that notify users when new information becomes available (BSCW,); technically, this is often realized using generated emails or RSS feeds (Hammersley, 2005; Board, 2006).

Discussion

CommSy represents a product in change—developed for a single purpose, simplicity became an important quality of the software. For mostly economical reasons, the use of CommSy has spread to other contexts, and while simplicity remains an important value for the developers, its meaning for the product has changed: it was necessary to add new functionality—new use contexts come with new requirements, and when there is a commercial interest, a feature will be added. As in the discussion of the Apple i-series, new versions tend to attract new features—even though CommSy is not sold as a product, but rather as a service. The need for new functionality is thus not simply caused by a need to create ‚more of the same', but rather by changing tasks and new users from different contexts.

Functional minimalism is slowly being replaced as an approach by structural and compositional minimalism, and simplicity takes on the meaning of ‚ease of use' rather than ‚sim-

ple functionality'; interviews with members of the development team indicate that ‚simplicity' still remains a key value, and that it is used in discussions as an argument both for and against the introduction of new features—a result of the gradual shift described here (see the discussion of the CommSy development process in 6.2).

For CommSy as one of two analyzed systems that were developed locally, the theoretical analysis was supplemented by an empirical evaluation of how the notions of minimalism qualify as analytic criteria. To this end, a survey with 69 students was conducted. Two groups of 53 undergraduate students and 16 graduate students differed by experience level, and rated CommSy's overall ease of use, as well as specific aspects such as sufficient and superfluous functionality, simplicity of access structure, the visibility of different tools in CommSy, and the support CommSy lent to their specific tasks, or the variability they saw in CommSy. The survey questions were deducted from the different notions of minimalism to find out whether the design would be uniformly rated on the resulting scales, and to examine the effects of rating on the different scales on overall ease of use.

The survey results show a large variance of answers, users specifically disagreed about whether CommSy consisted of tools that were termed ‚areas' in the questionnaire to accommodate the navigation metaphor. CommSy was not unanimously rated simple, by contrast, 36% indicated that it had too many unnecessary features (rating ‚too many features' as true with 4-7 on a 7 item scale). CommSy was rated as significantly easier to use by the group of students with less experience (scoring 5.6 vs 4.5, $p<0.05$). This result is possibly confounded with their use of the tool, which was mainly employed to distribute assignments and material needed to complete these, to a lesser degree to communicate important dates and continue discussions off-line.

In comparison with earlier analyses (comp. Janneck, 2006b; Strauss, Pape, Adam, Klein, & Reinecke, 2003), CommSy was consistently judged to be easy to use, the verdict, however, was less unequivocal. Looking for the reasons, a multivariate analysis revealed only two factors with a significant influence on overall ease of use: CommSy use was judged to be significantly less easy to use when the users believed it did have too many functions for their needs ($p<0.01$); the mean rating for ease of use fell from 5.8 to 5.2 and 4.5 on a scale from 1 to 7 as users were either indifferent about the number of features, or felt them to be too much. Another important influence on ease of use had the users' perception that CommSy did not hinder their individual work flow ($p<0.05$); here, users that felt constricted by CommSy only rated the ease of use at 3.9 in average, compared to a mean of 5.8 for all others. All other factors were at best weakly correlated with usability, one cause being the high variation in the individual scales, and the relatively low number of students in the graduate course.

A highly significant correlation was thus found between those users who considered CommSy to support different ways of working, and believed it to be ease to use, and those users who believed CommSy had too many functions, and found it more difficult to use. While the sample size is too small to draw further conclusions, the evaluation in-

dicates that it might be indeed possible that there exists a direct link between functional respectively compositional minimalism and usability.

Table 7: Minimal Aspects highlighted in CommSy

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: Again, relative to its competitors, CommSy limits the amount of accessible functionality, thus creating a comparably simple interface. Primarily during the foundation of the project, but also in current development, ‚Simplicity' is used as a value to guide design, highlighting the trade-off involved with the introduction of new features, often resulting in a careful and hesitant extension of functionality (some empirical evidence is provided in chapter 6.3, along with an example technique using values as guides).

Structural Minimalism: While other systems chose to follow consistency to desktop GUIs over structural simplicity, CommSy's unusual approach to storing documents in flat lists, enhanced by meta-data, displayed a high learnability, especially for non computer-savvy users. A compromise was made to create a high internal consistency—the structure to access functionality seems very similar between the different categories; yet, although the differences only appear at second glance, the network navigation differs with every information type, and the display of associations is carefully prioritized according to their usefulness.

Architectural Minimalism: CommSy is quite very visibly divided into different categories. The user can choose whether she wants to use a category, and the administrator can also hide categories not used. Categories could thus be considered „tools" that can be combined to adapt the CommSy system to a certain use context, partly by adoption, and partly by configuration. CommSy's categories work best when they provide clearly divided functionality. The category system, however, turned out to be not generic enough, or at least not applicable to all contexts, and categories were partially renamed for specific use contexts (e.g. schools).

Compositional Minimalism: Although CommSy's use domain was initially very specific, and included only a single type of University courses, its use has spread to other course types, Universities, and most recently also to other use contexts. In all these contexts, it is—often successfully—used to support communication. Apart from the dynamics created by value-based development (see 6.3.2), the functional minimalism, and the generality of the task of „exchanging documents" that CommSy fulfills are both responsible for the simple appropriation of the application. In some places, explicit abdication of functionality, e.g. access rights, has resulted in the replacement of an inadequate technical solution with an adequate social convention—and enabled the use of CommSy in contexts that it does not ‚support' per se. New use contexts create requirements for new features, threatening to undo functional minimalism as a quality (6.2).

5.1.4 Word Processing

In this chapter, word processing will consistently be used as the closing example as this is one area where for all four notions of minimalism can be related to specific mechanisms that were employed in real-world products. For functional minimalism, two projects are discussed here in more detail—both try to reduce the functionality that Microsoft Word or its competitor StarOffice offer by providing several interfaces that limit the available functionality.

Star Office 4 Kids

The first project, *StarOffice 4 Kids (Baumert, 2004)*, was initiated by Kippdata and the Heinz-Nixdorf Institute in Paderborn and is an example that shows both the potential and some of the main problems with functional minimalism: within the scope of the *Lernstatt Paderborn*, computer labs were installed at primary schools in town (Keil-Slawik & Baumert, 2004; Baumert, & Meiners, 2003b). To provide the pupils, most of which had little or no previous contact with word processing, with a tool that could be used to support learning (instead of replacing the object of learning), a simple alternative to normal word processors was developed. Using the software base of the open source project *OpenOffice.org*[19], a prototype was developed and deployed for a pilot study. Although the project also included a teacher interface (a slightly modified full version of StarOffice), the student's interface is more interesting in this context. Underlying the development was the proposition that fewer functions would allow children to quickly learn the basics of word processing; as new functionality was necessary for the tasks the teacher set, the interface would change and acquire access to new functionality with the addition of new buttons to the toolbar (Figure 47).

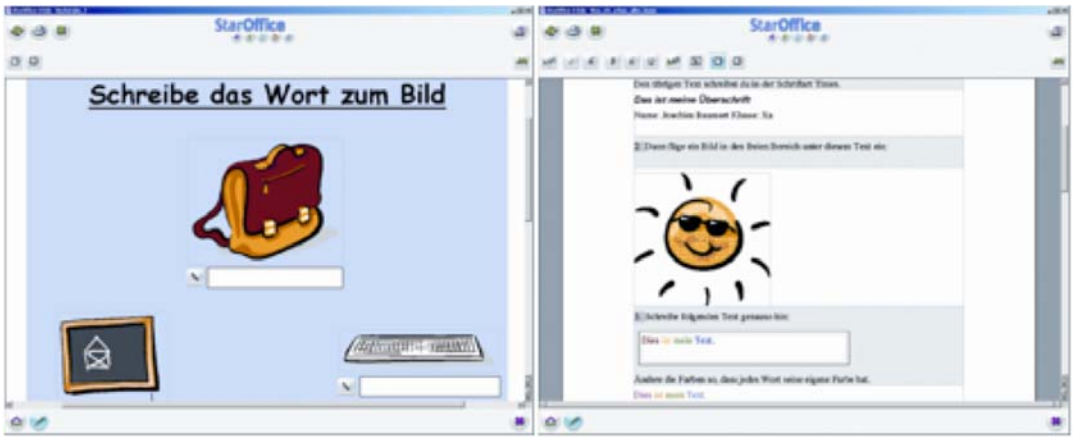


Figure 47: StarOffice 4 Kids prototype screens.

19. OpenOffice.org is an office suite that was released in 2000 into the open source after StarDivision, a small german software company, had been acquired by Sun Microsystems in 1999. Since then, it has been developed both by volunteers and by employees of Sun and became a well-known competitor to Microsoft Office.

StarOffice 4 Kids emphasizes the use of an open format, both as a distinguishing feature to Microsoft Word, and as a basis for allowing different versions of the software to access the same data. Although Keil-Slawik (2004) claims that different tools thus use the same file format, in this context the more important consequence is that the same document can be edited with differently limited interfaces. Used in conjunction with pre-defined tasks for teachers and students, this promises to provide some degree of seamless degradation of features.

In the evaluation report, Baumert (2003a) mentions that children had difficulties learning the meaning of the icons—a possible explanation is that they were not adapted to the school context from the original software, but used unchanged in the changed interface. As the teachers reported, the prototype was „too simple"—this was repeatedly mentioned in the interviews. It does not become clear whether the students were also convinced they needed „more" as interviews were only made with teachers.

The StarOffice4Kids example demonstrates how the direct implementation of the ‚less is more' idea can lead to mixed results. Implementing fewer functions in the initial interface lead to reports of a better learnability—although problems with icon recognizability persisted. Among the teachers, the acceptance of the new software was affected by their knowledge of the full functionality of a word processing application. They were missing features implemented in the existing de-facto standard, Microsoft Word. Although it is not clear whether they would actually have used these features, providing users with an interface that creates the feeling of being actively limited is not a promising approach. Another problem that was not mentioned in the report is that each new successive addition to the interface violated the internal consistency of the interface; it is very probable that this kind of extension will not scale up to the full functionality. The problems with icons might be partially rooted in the constant changes in their arrangement as their unstable placement prevented the effective use of spatial memory.

Evaluating multiple interfaces

As part of her PhD work, Joanna McGrenere examined the implications of providing different interfaces for word processing. To this end, Microsoft Word was altered using the internal Visual Basic for Applications scripting language, and a menu was added that could be used to switch between a minimal, a personalized, and the default interface (compare Figure 48). Both the minimal and the personalized interface were created by omitting menu items and toolbar icons.

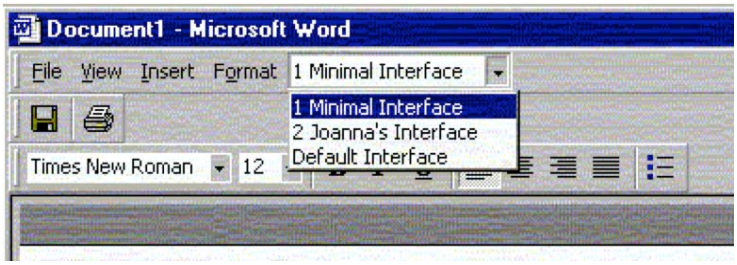


Figure 48: Multiple Interfaces: Menu to switch between the different levels. (McGrenere, 2002)

The second project was targeted not at children, but at ‚normal' professional workers: participants in the pilot studies included computer scientists, but also office staff. In contrast to the child users of the first project, these users did not have to learn the basics of word processing, they were adept users of the standard interface. As ‚expert' users, they were first asked how much functionality they really used. Only about 15% of all functions were used regularly in average, and only 12 of 265 functions were used often by more than three quarters of the users while 91 functions were used by not a single user within the sample of 53 participants (McGrenere, & Moore, 2000). This indicates that typically, dramatically less functionality is used than provided.

McGrenere concludes that the overlap of functionality between users is small enough to motivate the development of an personalizable interface. In a study with 4 participants trying out the three interface types, McGrenere found her personalized interface prototypes to be preferred to both the standard Word interface and the minimal interface (McGrenere, Baecker, & Booth, 2002) (compare Figures 49 & 50).

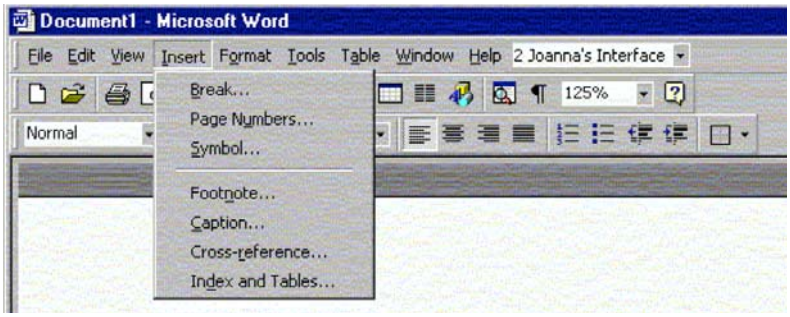


Figure 49: Multiple Interfaces: Insert Menu of the most complex personalized interface in the study.

What is most interesting in this context is that the minimal interface used in her first pilot study was preferred by one of two office users and one of two researchers, and continued to be used by one user after the study—as this interface provided less functionality than even Wordpad (comp. Figure 50). The minimal interface featured only 9 functions in toolbars, and had but a single entry in all menus but the File menu, where in addition to opening, saving, printing and quitting, the last 4 documents were listed (McGrenere, 2002, 248).

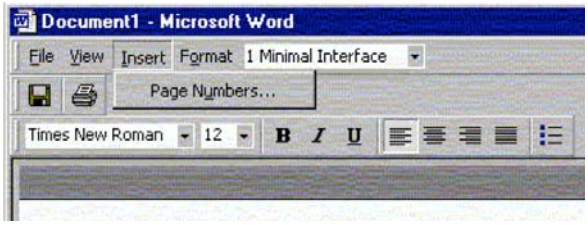


Figure 50: Multiple Interfaces: Insert Menu of the Minimal Interface.

The preferred use of this spartan interface indicates that Word might often be the wrong tool. This is only more surprising as McGrenere's use of minimal—"the 10% of the func-

tions from the DI [default interface] that are most frequently used" (ibid., 77)—demonstrates that for her, minimality does not require the remaining core to be whole; functionality is not balanced using real-world tasks. As functionality that is used infrequently can still be considered vital (e.g. printing), it is doubtful that the core functionality identified by a frequency-based method matches the core functionality for real world tasks. It is thus not surprising that participants expressed their preference for a less crippled interface—rather, it is surprising that the minimal interface was used at all.

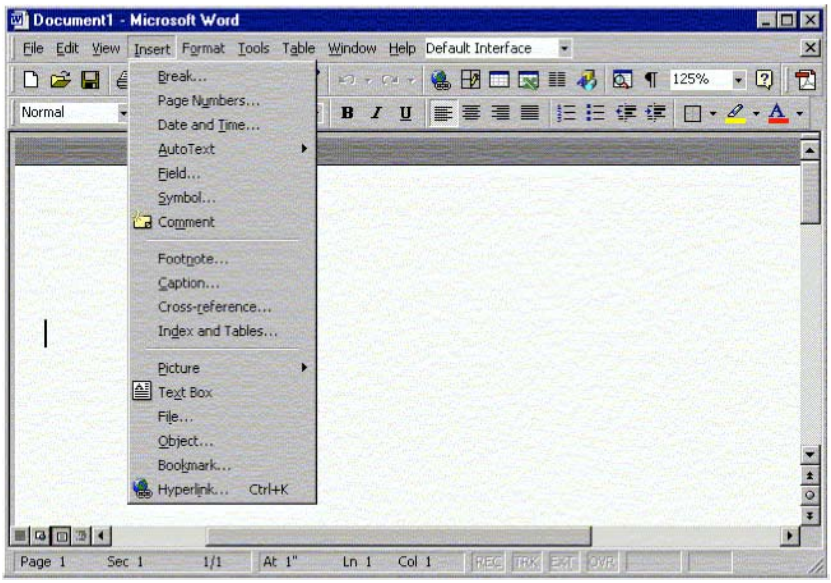


Figure 51: Multiple Interfaces: Insert Menu of the standard Microsoft Word 2000 interface.

The second word processing case demonstrates that standard office software, and Microsoft Word in particular, is commonly perceived as being overladen with features. Generally, participants expressed their liking of having less functionality, and for feature-shy users, McGrenere found an increase in the sense of control. The studies also suggest that it might be necessary to redefine the meaning of ‚functionally minimal' in personal terms—a simple dualism of novice and expert users is insufficient to describe differences between users: individual users use different functionality. A mandatory solution is not provided; what is missing is an extension on McGrenere's research—that focused on finding mechanisms for personalization—to a comparison with other methods of providing a best fit of functionality, e.g. a task-centric approach that tried to define a common, shared subset of functionality.

Discussion

The StarOffice4Kids case demonstrates that simply omitting functionality to make a product simpler will not necessarily yield a good design. As the required functionality grows with increasing mastery of the software, the use of new, more powerful interfaces becomes necessary—which introduces new inconsistencies. This highlights that functionally minimal designs create a need for tailoring, which can compensate the initial advantage gained by a deliberately primitive interface.

Tailoring functionality to individual users is the approach put forward by McGrenere. Although this might mark an ideal solution—everyone would be using a tool that would be completely customized to fit the individual needs—there are some problems with this approach: It is not always a cost-effective solution to have experts tailor interfaces to individual users; as with other tools, this still might be a viable solution for few high-performers. Studies have observed that specific employees in larger organization adopt this role as „tinkerer" (MacLean, Carter, Lövstr, & Moran, 1990), or „local developer" (Gantt, & Nardi, 1992). Yet, customization is not possible for all target groups, and will not always be efficient—there are trade-offs involved (Mackay, 1990; Mackay, 1991). Specifically, for cooperation between several users, customization effectively prevents cooperative reflection about the tool; users are hindered to help other users, use practices cannot spread easily (Grudin, 1994).

Both examples can be seen as examples for *layered interfaces* (comp. Shneiderman, 2003). Layered interfaces hide the full complexity of an application and expose partial functionality in layers. Each layer builds upon another, and exposes more functionality. While StarOffice4Kids closely follows the layer concept, adding functionality sequentially as the learner progresses, McGrenere's layers in her multi-interface are partly caused by the study design that aims to test the concept of a personalized interface. The benefits for expert users are unclear as better learnability is not a vital factor for them. If reduced interfaces are useful even for expert users as they perform less demanding tasks, this points towards severe deficiencies of the original interface that harm the user's performance.

The concept of layered interfaces is not undisputed—although it is put forward as a strategy to further universal usability (ibid.) and simplify the initial learning curve (Kang, Plaisant, & Shneiderman, 2003), it has not turned into a movement extending beyond Shneiderman's HCI laboratory, and the „graceful evolution" of interfaces (Shneiderman, 2003) can be problematic. Earlier research indicated that non-layered software might provide a better basis for seamlessly learning advanced features of an application. Nardi stated in 1990 (1990a) that Microsoft Excel allows for a multitude of different uses at different levels of proficiency. This can be generalized to other spreadsheets—the table model has been identified as a conceptual basis for the discipline of end-user programming (Nardi, & Miller, 1990b; Nardi, 1993; Myers, Weitzman, Ko, & Chau, 2006; Myers, Ko, & Burnett, 2006).

While support for end-user-programming is from a user's perspective to be welcomed[20], it is not a good design solution per se. Rather, it could be named an indicator of defeat—as the designer does not know what features are going to be used in which conjunction, he settles for providing a toolkit that the user then has to pick the useful tools from. This represents an antagonistic position to a functionally minimal design. Mass market design is generally unable to depend on users to tailor the design their individual needs, and tries

20. Other perspectives, e.g. that of the „professional programmer", tend to fear user programmers for their well-adapted, but poorly engineered code which may create severe security problems (Harrison, 2004).

to find a common core of functionality—which steadily grows as new users are examined, a certain factor driving the feature frenzy.

5.1.5 Refining the Notion of Functional Minimalism

Judging from its definition, *functional minimalism* is the most trivial form of minimalism and can be immediately measured: the less functions a design has, the more minimal it is. However, the discussion of actual products has shown that (1) minimal functionality is closely connected to other forms of minimalism, (2) defining what is minimal is far from trivial, (3) definitions of what is minimal are subject to change as tasks change and new design revisions are based on previous designs, or (4) new groups of users are targeted, exemplified by the CommSy system. As the analysis of word processing applications demonstrated, a form of functional minimalism can be achieved by (5) layering the functionality in several interfaces that the user can progress as she learns to use the program, or choose according to task.

Both the Commsy developers and Apple designers tried to achieve a reduction in functionality through a shift in responsibility. Their designs do not try to provide all necessary functionality, but rather limit themselves to a basic set of functions. As both designs target users who are more concerned about the ease of use of their tools than about their comparative power, or have other, more sophisticated tools in their toolkits, the design choice is met with approval. Even here, however, the consequences of the *2.0 syndrome* become obvious—subsequent versions of the software tend to accumulate functionality, blurring the difference between the new tools designed to be clean and simple, and their more complex competitors.

Word processing applications face a more direct dilemma. As their producers fear to alienate potential users, the functionality considered as belonging to word processing has grown to an enormous amount of different features. The question of how much functionality is really necessary gives room to a dynamic that requires new functionality in each new release. An empirical indication that new functionality can actually harm performance was delivered in a study by Franzke and Rieman (1993) with 12 users who had an average of two years computer experience. Two different versions of a graphing package were used to create a default graph. The task was performed significantly faster ($p<0.05$) using the earlier version of the package than the later version[21], demonstrating the possibility of a negative performance impact of new features.

Limiting the functionality of an application to support use on different levels is not a new concept in itself. In the 1980s, John Carroll and his team created the *Training Wheels Interface* for a commercial word processor, an interface that blocked some functionality, displaying a message when a blocked function was accessed. Two studies with 12 novice users each compared the training wheels system to the complete system. Results of the

21. Task design plays a role here; if the experimental task had required the use of features available only in the newer version, then using the newer version might have been faster, or a comparison might have been impossible.

first study indicated that users could complete a simple word processing task about 20% faster with the blocked interface. More importantly, they spent significantly less time recovering from errors[22]. Comprehension of word processor basics was significantly better for users of the blocked interface, as was their score in a questionnaire designed to reveal users' attitude towards their work. Results of the second study were almost identical, lacking only significant differences in the tests on comprehension and work attitude. While blocking of functionality is still used, and has been promoted as a *layered approach* by Shneiderman (2003), who suggests that novice users should start with a minimal subset of functionality to protect them from making errors, and progressively learn more and more advanced features of the software, its usefulness for expert users—and all users are bound to lose their novice state—is not clear. It is also not trivial how to design a layered interface, and whether there should be more than one path of progressing through layers. Shneiderman doesn't suggest specific design guidelines to accomplish this design strategy.

5.2 Structural Minimalism

When functionality is more than minimal, and a tool serves more than a single purpose, a selection of functions by the user must be possible. The a*ccess* to the functionality of a device is structured by the designer—she chooses which knobs, buttons or menus to provide, in short: she designs the *interaction structure*. Using the notion of minimalism, an access structure can be considered more minimal as it is perceived to a lesser degree by the user. That is, while the size of the structure (and the number of menu choices or buttons) can be fairly large, the required functionality must always be ‚at hand', and no „navigational excise" (Cooper, & Reimann, 2003, 135) be required from the user.

5.2.1 Remote Controls

As a real-world example for dealing with access structures that is known and feared even by those who are hesitant to use a computer, home entertainment systems may be a good example for the difficulty of using technology: as is widely propagated, few people, and even few computer scientists are able to program their home video recorder (e.g. Norman, 2002b). Looking at the remote controls that ship with a current DVD player (Figure 52), an intimidating display of functionality threatens to allow the effective use of the device—few people will need all the functionality that the multitude of buttons can set in motion.

22. The experiment defined error as deviation from the ideal action path, and recovery as steps in the subsequent return to that path.

Figure 52: Selection of some remote controls for contemporary DVD players.

By contrast, Apple designed a minimal remote control that concentrates on the key functionality required for a home theatre, namely play and pause, skip and search (Figure 53). While this remote could also be classified as a functionally minimal at first sight, the menu button in conjunction with the (then) direction buttons allows the execution of more advanced functionality by navigating a menu system.

Figure 53: Apple Front Row remote.

Bang & Olufsen, a Danish company producing high-end Hi-fi technology, tried a similar concept in the late 1990s with their remote control *Beo 1*, launched as an alternative to the *Beo 4* (Beocentral, 2005) (see Figure 54). Both superceded the *Beo 5000* that isolated infrequently used functionality on the back of the remote; while the *Beo 4* was a conventional universal remote control—its modes (that relate to the devices in the stereo set) determine key mapping, the *Beo 1* was an attempt to minimize the number of visible buttons. As their designers state in a press release: „None of its buttons are numeric or source related, instead the operation focusses upon an intuitive interaction with on-screen display. It takes product control to a new transparent level where operation becomes a part of the total experience." (Bang & Olusen, cited after Kiljander, 2004, 160).

This concept proved to be unsuccessful, as the navigation was only displaced to a screen interface. This required the television to be switched on continuously, and also provided only a poor alternative to directly mapped keys; timeouts were applied by the interface so that „key press sequences felt even longer to the users" (ibid., 160). Fortunately for B&O, the *Beo 4* used compatible IR signals, and the experimental *Beo 1* was dropped silently.

Figure 54: Bang & Olufsen remotes: Beo 5000, Beo 1 & Beo 4.

Another possibility for interpreting minimal structure is represented by the harmony remote control that was reviewed by Don Norman (2004a): as B&O's Beo 4, it is a remote control that tries to replace several other remote controls. But it is not simply reprogrammed to make available all functionality that the individual remotes offer to the user; it rather groups them according to activity—a single keypress can switch on the television set, dvd player, and stereo, and reroute all input signals so as to enable the activity of „viewing a DVD" (Figure 55).

Figure 55: Harmony remote control.

This instance of ‚activity-based design', so Norman claims, greatly simplifies the task of coping with expensive electronics in the home and makes for satisfied users. However, the

approach has limitations as problems arise quickly if the system's prioritization is not similar to the users (for a less satisfied customer, see DeBoer, 2004). The activity-centered design illustrated here adds a new layer of understanding to structural minimalism: not only can atomic functionality be ordered according to task demands, but new atomic operations—that may even span several information systems—can be composed; if the closely match the user's actual activities, an increased sense of simplicity is created.

Discussion

Structural minimalism was defined as the access structure to functionality that presented the minimal perceived structure. This does not equal the minimal amount of structure, but rather forces the designer to think about how his layout of functionality will be perceived. The remote control example shows that mindsets are very different when it comes to structuring complex functionality. There are those who believe that every function should have its physical counterpart—and are bound to fail as the number of functions grows faster than the space on remote controls. There are those who reduce the physical interaction to a minimum, trading off increased complexity in menu systems; whether users accept this shift seems to depend on the application domain. A possible solution for some applications might be the combination of individual functions into larger scripts, and the contextualization of commands. However, this approach will only work where function sequences are often similar, and contexts can be clearly identified.

5.2.2 The Palm Handheld

There have been few commercially successful revolutions in the design of interactive systems since the Graphical User Interface „desktop" metaphor has been widely accepted. One notable exception is the introduction of the Palm Pilot in 1996, an extremely useful information appliance that had very limited functionality but happened to work in practical environments. In contrast to its more elegant and powerful but less useful predecessors, like the Apple Newton, it won the market and created a new way of computing.

Palm Computing started off very quickly, selling almost half a million units in the first seven months and quickly gaining dominance in the hand-held computing market. Currently, for 2004, the market volume for handheld computers is estimated to be around 11.3 million units. This immense market did not exist prior to the introduction of the Palm Pilot, the first palm-sized organizer. But the market has diversified, today the Palm series has to compete against the Windows CE handhelds, generally equipped with better displays, faster processors, and better networking capabilities. Despite these unfavorable engineering figures, the Palm series sells very well. It still draws on the qualities that enabled its success and fights with difficulties because changes to the original concept always carry the danger of losing its outstanding position. But what exactly are the qualities of the Palm handheld? Although economic factors such as the acquisition of Palm Computing by US Robotics made the Pilot's success possible in the first place, the reason why exactly the Palm succeeded not only over his competitors but also in forming a new market segment is to be sought in his superior usability and practicality.

Figure 56: Palm Pilot 5000 (1996), an early model.

Description

A number of things Palm carry the notion that reduction was the key to success. In an unusually open interview, Rob Haitani, a former Sony employee in charge of the Palm UI development, claims „every pixel counted", that giving up three-dimensional button emulation and the reduction of font size were important to the success of the product. (Bødker, 1990) Jeff Hawkins, the founder of Palm, was said to walk around with a block of carved wood in his pocket, a prototype smaller than the PDAs at that time, to demonstrate that size and features had to be reduced to achieve greater portability.

However, as the perspective of time shows now, these were solutions to immediate limitations that the Palm faced during its conception phase: They made it possible to construct the device in its time, allowed useful information content to be displayed and processed – an example of brilliant engineering – but today they don't give the Palm series a principle advantage anymore as processor speeds, screen resolution and color depth keep increasing. Instead, the advantage turned into a disadvantage when, during the introduction of high-res Sony handhelds, no accepted standard was available; even the newest Palm OS draws little benefit from the increased pixel count for the sake of compatibility.

Previous attempts to create a handheld computer were commercially unsuccessful. The most prominent example was the Apple Newton: it had a very sophisticated interface, yet one of the primary functions, the handwriting recognition software, was severely lacking functionality when the Newton was first released. In contrast to other companies, Palm did not believe that ‚the first-generation handhelds failed because they did not provide enough functionality' (Bergman, & Haitani, 2000). The Palm was less ambitious, and tried to concentrate on the essential functionality that an ultra-portable computer needed. The design goal was to create a small, fast and inexpensive device.

As the traditional crafts differentiate between numerous types of tools for different tasks (e.g. hammers for hitting nails, paving stones or knees), and multi-function-tools are more often sold in do-it-yourself stores than successfully employed at work, it is attrac-

tive to transfer the concept of single-purpose tools to interactive systems. Don Norman's (1989) notion of *Information Appliance* is commonly applied to the Palm—and even to the functionally more sophisticated Blackberry (Shedroff, 2001). While this seems fitting at first, Palm Pilots not being general purpose computers, it is misleading: the Palm could always do more than one thing. It was designed to replace the business traveler's organizer, an analog physical tool with many functions. Precisely this combination of functions promises an added value of the electronic version, as cross-references need not be maintained by hand, and copies for backup or communication are faster and cheaper.

In the Palm Pilot, we find a high degree of structural minimalism, as a number of metaphors common to the desktop world have been replaced by a simpler internal structure that is transparently conveyed to the user. Design penetrates the system: Palm did not just design a user interface; the whole architecture is designed to fit the requirements. The initial versions of Palm OS allowed only one application to run at a time, introducing a simple mapping between what the device was doing and what it would display on the screen: what you see corresponds directly to the internal state. Likewise, the direct access to applications provided by four buttons on the lower front of the Palm case represents again a direct mapping, this time from input to state.

Because of this ease of switching between applications, the necessity of preserving changes arose. In contrast to almost all the known computing world, the Palm Pilot rejected the notion of files. There is no save; there is no file system (but for an invisible flat database store), so there is only the current state that will be automatically preserved. This simplified model of dealing with data has been present in Apple's Lisa, but since then operating systems forced users to differentiate between memory and disks, volatile and permanent information.

It is interesting to note that the design concepts for the Palm were not developed upfront, as Haitani says: „It was more an end result of our pragmatic design approach, starting with the fact that we could only fit four buttons on the screen." (Bergman, & Haitani, 2000) By rigorous user testing, the most often used commands were optimized for performance. Haitani even developed something his fellow workers called religion—„minimal click counting" (Butter, & Pogue, 2002): the ultimate goal for designing a given functionality was to reduce the number of stylus movements for the user. The result was an unconventionally structured, yet rapidly usable interface.

The Palm device was specifically designed as a business tool—in contrast to today's often Windows-based PDA, it had a different look-and-feel, and none of the applications common on desktop computers were available on the Palm, e.g. there was no functionality to view or edit office documents. The number of existing applications was extremely limited: a calendar, an address book, and a notepad. Each of these applications was designed with only minimal functionality as ‚most people only use a small percentage of the features in an app' (Bergman, & Haitani, 2000). Still, the combined use of these applications can create a powerful tool supporting complex tasks. Specifically, the integration with the „Palm Desktop" on the desktop computer allowed a synchronization of events

and notes, and thus enabled a division of labor: the Palm was used ‚on the go', as a tool in meetings and for phone calls, while the desktop computer handled email and document management with its larger display and faster input channels. The Palm was a supplement that could do things the PC could not. Strictly speaking, this disqualifies the Palm as an information appliance—its integration and ability for cooperation with a PC was one of the key features for its success.

This *architectural minimalism* created, however, difficulties when Palm was later forced to compete against Microsoft-powered PDAs. These PDAs offered more functionality to the broadening user base. As a reaction, key people at Palm left the company to form HandSpring and take the initial idea to the next level: The „SpringBoard" was to extend the essential functionality of the Palm where needed, a byproduct of Palm's strategy of supporting only the minimal set of functions and also the beginning of competition with other devices. This venture, however, proved to be more difficult than expected. Apart from reasons as the difficulty of transforming a software company into a hardware supplier or the limited financial resources, the development of a proprietary slot for seamless plug and play made the hardware expensive, expansions often matching the original device in price. While the initial PalmPilot provided basic functionality for everyone, HandSpring would provide specialized features for those who sought them – too few in the end. Finally, the company was bought again by Palm and hope for Palm's future still relies on their latest development, the integration of mobile phones and the Palm device.

Discussion

The Palm is a straightforward example for the rigorous application of usability methods to attain a minimal access structure. For the sake of structural minimalism, even internal consistency was sacrificed—menu items that would traditionally have been grouped together are presented in different layers, and as the Palm is for minimal clicks, every application behaves slightly differently. Although the criterion of consistency was violated so rigorously, the Palm felt very easy to use; so much was stated in all initial reviews. Instead, a key component of the Palm's success was its compatibility: because the exchange of data between standard PC applications and the Palm was so easy, the inconsistencies did not matter. After the initial niche was occupied, and more functionality was sought, Palm lost its advantage – as other PDAs directly mimic PC applications, the mapping of extended functionality is much easier. And while the ease of handling may decrease, consistency with known applications became more important than optimal adaptation to the task—the lack of external consistency became an important negative factor for Palm sales.

Structural minimalism as defined here can be pursued with common HCI techniques. The development of the Palm Pilot was largely carried out using prototyping[23]. Using real or simulated tasks, the interface was adapted to small tasks, trying to optimize those tasks

23. This begins with the anecdotal wooden prototype that the Palm founder, Jeff Hawkins, carried around to show the form factor of the future device, and is most visible in the many iterations that were used to optimize the click (or rather tap) interaction with the Palm applications.

that were carried out the most often. This resulted in a smooth and intuitive handling. By contrast, PDAs are now developed and sold by the number of features, technology such as WLAN and G3 mobile communication connectivity, integrated cameras or voice recorders has become more important that the basic functionality. With competition from mobile phone companies and major desktop software vendors the direction that development will take is still not decided (Lien, 2001; Gartner, 2004).

Again, a *2.0 syndrome* limits the success of a minimal design. The picture is even more dramatic; by introducing a new computing paradigm with a minimalist device the demand was immediately created to expand the initial, minimal functionality. This is why the Palm series is losing ground today – even though most people still need only basic functionality. Extending a minimal device is a challenge and whether selective addition of key features can outweigh the marketing power of missing features is yet to be decided. The convergence or diversification in the mobile computing domain remains an interesting subject of study.

Table 8: Minimal Aspects highlighted in the Palm handheld.

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: Although the Palm is the most prominently used example for an *information appliance*, it is not functionally minimal in the sense of focus on a single function. Rather, its field of competence is clearly defined by excluding those areas of use a PC is better suited for, thus limiting the duplication of functionality. The orientation on this use setting created a minimal, yet whole set of necessary functions.

Structural Minimalism: The „counting clicks"-technique in association with iterative prototyping has made the Palm the most convincing example for the interpretation of *structural minimalism* that tries to reduce the visibility of access structures by optimizing the structure to fit to the probability with which a function is needed. This is not limited to menus, it also extends to information layout, or automatic modes that are chosen e.g. when the notepad is opened, or the calendar selected—functionality in contextualized, trading off immediacy for (internal) consistency. The combination with the relatively minimal functionality, and the focused area of competency allow this approach to create a very simple seeming, intuitive device. Problems became visible as the functionality needed to be expanded.

Architectural Minimalism: „The Palm" is actually more than the hardware device. Only the close integration the free Palm Desktop builds with PC applications generates the value in using a Palm: all changes—whether on the PC or the Palm—become immediately synchronized with a single button press, and data is automatically forwarded to third-party applications, most notably Microsoft Outlook. The close integration frees users from having to remember where they stored notes, or appointments, and creates the illusion of using two tools within a single system.

Compositional Minimalism: The access structure of the Palm was optimized for very specific tasks, and the functionality tailored to the exact needs of a business user. Some functionality was missing, however, as users despite the form factor wanted to use the Palm to e.g. edit office documents. The Palm could be programmed to adopt it to other tasks, and some applications were developed for other contexts, but the unique design that proved so beneficial for its initial fit to the task made it inferior to Windows CE handhelds, where applications could build upon the consistency with existing PC applications.

5.2.3 Minimal Access Structures for Mobile Communication

Mark Weiser is often referred to as originator of the label of *ubiquitous computing* (Weiser, 1991; Weiser, 1993; Weiser, & Brown, 1997). This label describes research directed at the interface of computers becoming invisible, and includes a vision of a tight integration of technology with everyday life. Regarding information appliances (Norman, 1999a)—integrated computers that exist in e.g. washing machines or electric drills, mobile phones have perhaps had the most fundamental impact on daily life[24]. In contrast to other visions, e.g. designing for smart homes, domestic use and smart environments, in mobile phones computers have actually managed to „vanish into the background" (Weiser, 1991) to some degree.

Description

An approach adhering to structural minimalism for mobile phones must tackle the difficulty of providing an access structure that seems to be simple, yet allow access to the manyfold functions of today's mobile phones. Traditional phones used to feature a numberwheel (Figure 57), or a numerical keypad.

Figure 57: Analog phone dial.

24. Continental Research conducted a study for Vodafone UK in 2002 indicating that half of British business travelers state that their mobile phone is their most important possession on a business trip—more important than clean underwear, a razor, or toothpaste (m-travel, 2002).

During another study by Codacons Italy in 2001, 300 volunteers were parted with their mobile phones. 15 days later, 70% reported problems including loss of appetite, sexual problems, depression, and a general blow to their confidence (Batista, 2001).

Additional functionality, such as repeated calls, listing received calls, or quick dialing of stored numbers, was typically mapped to new keys that were added to the phone's interface. For mobile phones, the problem became much more complex, as new functionalities like SMS exchange were integrated with the phone, and the internal status—settings for ringtones, vibration alarm, provider selection, address book management to name but a few—created new tasks in itself.

Facing the difficulty of providing simple access to this complex functionality, Nokia used an approach very similar to the click-counting of the Palm developers in the commercially very successful 3100-, 3200- and 3300-series of mobile phones (Kiljander, 2004, 80); these phones were targeted directly at the mass consumer (Karjalainen, 2003), and little tolerance for complexity was assumed during design. In addition to a standard numerical keypad, they featured a *Navi-Key*, a ‚softkey' whose semantics changed according to the user's actions—or rather, whose functional mapping changed according to the system's state (comp. Kiljander, 2004, 109ff). The function that a key press initiates is determined based on the frequency of functions chosen from the current system state. Although other functions can be reached using the scrolling keys, this is often not necessary. Thus, the interface (compare Figure 58) was reduced to include only the Navi-Key, scrolling keys, and a clear (undo) key (Lindholm, & Keinonen, 2003, 24); the intended message was „Anybody can master this phone as it is operated with only one key" (ibid., 25).

Figure 58: Nokia 3310 phone.

In usability tests, a preference was noted for phones with the Navi-Key as perceived usability (comp. ibid., 25) was higher than both for the older 2110 series and current mobile phones (ibid., 85). Ziefle (2002) compared the Nokia 3210 interface with a Siemens C35i and a Motorola P7389, and test users showed higher performance with the Nokia phone. Bay & Ziefle (2003) furthermore compared the menu structures of the C35i and the 3210, and found that for the Siemens phone, menu structure was significantly more complex and use of control keys was significantly more difficult than Nokia's—users took twice the time to complete tasks and made thrice as many detours in the menu[25]. Finally, Kil-

25. It should be noted that the result of this usability test cannot be traced to a single UI element, i.e. the Navi-Key might contribute to, but is not solely responsible for the better performance of the Nokia phone.

jander (2004, 191) found that Nokia Navi-Key users rated the new interface of the Nokia 6650 as „less easy to use" compared with their old phones. Interestingly, the Nokia designers themselves differentiate *actual* „ease of use" and *perceived* „ease of use": for the actual usability, the total number of keystrokes necessary to select a function is authoritative. Thus, by their definition, a reduction of keys decreases the actual „ease of use", while the perceived usability is increased (comp. Lindholm, & Keinonen, 2003, 25).

Yet with all its merits, the Navi-Key approach has some drawbacks: Mapping the Navi-Key to the most often used function in a certain situation will not work for all users—Nokia nonetheless chose not to implement a learning algorithm in their phones. Access structures were slightly reordered in the development from the 3210 to the 3310 phone, but no adaption mechanism created an individual structure that would optimize access for an individual user. Adapting menu structures is known to cause problems due to a lack of consistency (Mitchell & Shneiderman, 1989; Tsandilas, & schraefel, 2005). In an interface where a single function is mapped to an element, system reactions are very predictable (see 4.4.3). As the phone menu interface does not provide a history of menu selections, error recovery caused by assumed consistency becomes very difficult. Furthermore, the menu system, and with it the conceptual model for phone usage, is still technically oriented, e.g. regarding dual memory (Klockar, Carr, Hedman, Johansson, & Bengtsson, 2003): messages or numbers are treated differently according to their storage location (SIM-card or internal memory).

The culmination of minimal structure was implemented in a different market segment: luxury phones, such as the Vertu concierge line[26] are bought in combination with a service contract. This ‚concierge' service is contacted with a designated button—a human answers the call, and any further function, ranging from looking up a number to booking a hotel room, or ordering tickets, is then executed by this human agent (Nokia, 2002). If nothing else, this non-technical solution points towards limits of our interaction with technology.

Figure 59: Vertu concierge phone.

26. http://www.vertu.com

Discussion

Again, the Nokia 3110 phone follows the ideal of a *minimal access structure*, and again, the result is reviewed favorably. The design of the Navi-Key is the result of a simple design technique, the selection of functionality based on use frequency. The single-key concept performs surprisingly well, but does not scale to newer Nokia models that feature more functions: here, the single Navi-Key has been replaced by two reprogrammable keys.

As new phones acquire new functionality, their technical innards become more visible for the user—the introduction of technobabble like WAP, GPRS, MP3, Bluetooth etc. that is little understood by end users (Metafacts, 2003) into phone ads may suffice as proof—and the computer within the phone surfaces again. Although terms such as „Simplicity" or „Ease of Use" are featured prominently when mobile phones are advertised (RPO, 2005), usability studies indicate that phones are increasingly difficult to use (Dahm, Felken, Klein-Bösing, Rompel, & Stroick, 2005; eco, 2005), and that consumers are overwhelmed by technical features (de.internet.com, 2004); especially for older adults, the use of ‚advanced' technology becomes problematic (Ossenbrügge, 2004b; Ossenbrügge, 2004a). Mobile phones are still primarily used for tasks the Nokia 3310 can handle, voice and SMS messaging—the need for sophistication of mobile phones is thus not obvious. However, the *2.0 syndrome* here is driven by the telecommunication companies trying to find the next killer application that will help pay for their high-speed networks.

The Vertu concierge line underlines that even ‚user-friendly' interfaces are inferior to the speech-based interface of person-to-person communication. HCI has not been able to come up with an alternative to this really intelligent interface, yet. Just as the reduced number of buttons on remote controls, the single button on the Vertu phone is no technical solution, it only shifts the users' problems to another, more friendly modality.

Table 9: Minimal Aspects highlighted in Nokia's 3310 series

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: At first glance, the Nokia 3310 line of mobile phones limits itself to the core functionality associated with phones: voice and messaging. This is, however, partly confounded with the age of the phone: contemporary low-cost phones targeting the mass market have a similar range of functionality. As the phone's deep menu structure proves, much of the functionality, e.g. for programming ring tones or configuring network services, is accessible, alas well hidden.

Structural Minimalism: This seemingly simplistic design of the Navi-Key is a prime example of a probabilistic context-based access structure. As in many situations, a single function is chosen with extreme preference (e.g. ending a call while talking, or reading a message after receiving a notification), the „one-key" solution is often able to suggest the „right" action. As the minimal remote controls, the Navi-Key also has a communicative function: it suggests visually that this phone is simple to use. As an important aspect of *structural minimalism* is that the design is in itself not truly simple,

nor offers only minimal functionality, this could be interpreted as deceitful, and the simplifying effect could be reversed if the deception becomes obvious. Later variations of the Navi-Key concept that use two or more „smart keys" demonstrate—although the concept can be extended by adding more probabilistic buttons preventing the more sophisticated functionality to become hidden by the mundane, often used features—that the clear mapping between feature and physical interface is lost through the addition of functionality.

Architectural Minimalism: As the 3310 series was primarily designed as a stand-alone device, its design does not imply its use as a ‚tool'—only the infrared connection that allows synchronization of event reminders might be considered as a link to a Microsoft Outlook calendar application.

Compositional Minimalism: Nokia's concentration on voice and short messaging services have created a simpler device, but there is no evidence suggesting that this increased the use of the 3310 series in unconventional domains. On the contrary, specific types of phones are designed for e.g. with four buttons as an emergency phone for the elderly, or with parental budget control for children. The Vertu concierge series has implemented a Wizard-of-Oz-like variant of the ideal of an intelligent phone, relaying the intelligence into the human operator; although this creates a universal tool, it does not generate design advice—except that it may sometimes be adequate to accept limitations of technology, and find a social, rather than a technical solution.

5.2.4 Hyperscout: Enhancing Link Preview in the World Wide Web

Wherever you go, there you are.
Buckaroo Banzai. Rockstar, neurosurgeon, inventor, movie character

Links are the most prominent medium for interaction with the World Wide Web, and thus should be listed among the most important interface elements of today. However, the interface of links—their graphical representation and the actions afforded—has not been a field of intensive research. Rather, the design of hyperlinks has been inherited from the first accidental renderings in early browsers (Weinreich, Obendorf, & Lamersdorf, 2001), and the fact that small changes in the appearance of link markers can dramatically change reading behavior (Obendorf, & Weinreich, 2003) has been largely ignored.

As disorientation has repeatedly been stated as one of the major problems of navigation in the Web—for Hypertext systems, Conklin took note of this problem even in 1987 (Conklin, 1987)—the *enhancement of the link interface to better support navigation* is an important but underestimated problem. The Hyperscout project (Weinreich, Obendorf, & Lamersdorf, 2004) is an effort to ease navigation in the World Wide Web by providing link preview information.

Technically, it is based on IBM's *intermediary* concept (Paul & Rob, 2000), and the Scone framework (Weinreich, Buchmann, & Lamersdorf, 2003): Web pages requested by the user are filtered by the intermediary, allowing the addition, subtraction, or modification of all data. Hyperscout adds information to links in Web pages—information about the

target of the link; this information is collected invisibly in the background while the page is being rendered and displayed in tooltips when the user moves the cursor above a link marker (compare Figure 60).

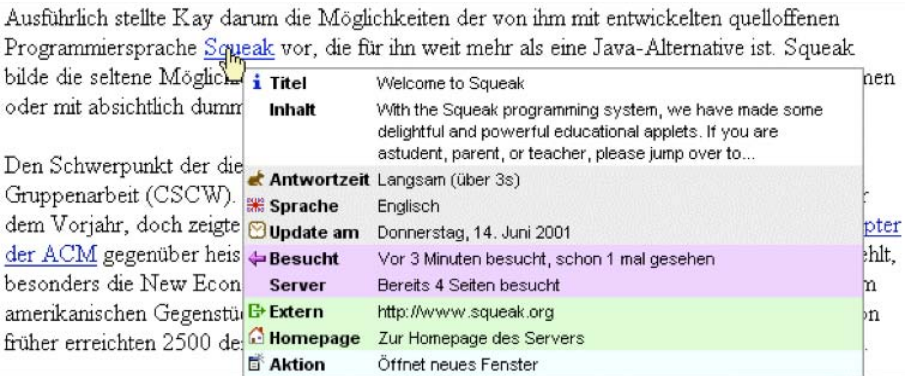


Figure 60: Hyperscout popup window displaying meta-information about the link target.

An important aspect of the design of Hyperscout is the selection and granularity of preview information. To actually support navigation, it is vital that all the displayed information is useful, but also that the displayed information is sufficient—it must be minimal but also complete. If the information satisfies these needs, the user is freed from „navigation excise" (Cooper, & Reimann, 2003, 135ff) as she is empowered by the additional information to better judge which link is leading to the desired information.

Description

In terms of minimalism, Hyperscout is untypical as it explicitly adds information to Web pages. This is necessary as although the Web itself is abundant with information, users are provided insufficient means to cope with the information overload that has become a hallmark of the information society (Wurman, 1991; Berghel, 1997a; Farhoomand & Drury, 2002). Hyperscout follows the proposition that additional information about hyperlinks will help users assess the importance of the linked information, thus enabling a fair judgement whether it is necessary to follow a link. The expected benefit for the individual user is a reduction in visited pages as detours are reduced, combined with a lower cognitive load as link meta-information is provided by the system and does not need to be kept track of by the user.

Often, type information for links is referenced as the most important information source, as it would in principle allow a detailed analysis of link usefulness (comp. Weinreich, Obendorf, & Lamersdorf, 2001). It is, however, rarely provided in the Web, dramatically limiting its usefulness. Often, the technically trivial thumbnails are used in as preview information in research prototypes. Although they look impressive, they are unfortunately of little practical use as the content is rendered illegible and many sites use a corporate design that prevents the identification of individual sites in thumbnails[27]. Web design

27. Some approaches use thumbnails for link preview (e.g. Kopetzky, & Mühlhäuser, 1999; Nanno, Saito, & Okumura, 2002); this was also implemented using the Scone framework (Wollenweber, 2004), but was

conventions that unify e.g. navigation menus across sites do not help to distinguish their visual appearance.

There are different types of implicit meta-information: Weinreich et al. (2004) list contents of the target document, link topology, use history, media type, browser action, target status, and expected reaction time as information that is implicitly available in the Web. Using this existing information that only has to be harvested in advance, a detailed picture of the target document can be created without having to navigate there. To actually effect a lowering of the cognitive load, a vital part of the Hyperscout system is knowing which information is useful in what situation.

The amount and type of useful information depends on the status of the linked page. For example, the necessary information for a non-existing page is very different—Hyperscout indicates the link as broken (compare Fig. 56)—than the information for an existing page. For certain types of pages that are updated often (e.g. news sites), the number of changes is more interesting than the last update; the number of visits to a page is especially important for long-term revisits—a large number of heuristics can be identified that will lead to a choice and ranking specific to a certain use case.

An important aspect of the link interface is the visualization of link markers within the currently read document. From the many existing possibilities (comp. Noirhomme-Fraiture, & Serpe, 1998), underlining is the de-facto standard—not the result of a conscious design process, but a historical heritage—and other alternatives have not been successful (Weinreich, Obendorf, & Lamersdorf, 2001). Only in the recent past, a trend towards simplification—omitting the underlines, leaving only color to mark links—can be observed in the Web; a trend that increases the readability of Web pages (Obendorf, & Weinreich, 2003).

A challenge for the Hyperscout system was thus to make additional information available to the user without increasing the complexity of the layout. While icons are often proposed to add preview information[28] (comp. e.g. Hightower, Ring, Helfman, Bederson, & Hollan, 1998; Campbell & Maglio, 1999), informal tests with the Hyperscout system showed that the textual form of meta-information was often superior in ease of understanding. As the textual information could not directly be added to the Web page, pop-up information windows were chosen to display link meta-information—just as context menus allow access to functionality useful to manipulate the current selection (Koved & Scneiderman, 1986), these pop-ups display contextualized information providing useful information about the link under the cursor. This follows the perspective of *structural*

found to be not very useful in user tests.

28. In previous versions, Sun's Wb design guidelines (Levine, 1996) and Yale University's Web Style Guide (Lynch, & Horton, 1999) included hints to decorate external links with icons. Parts of the Microsoft web site also follow this practice (e.g. Windows Hardware Developer Central at http://www.microsoft.com/whdc/Legend.mspx), and plugins for CMS systems allow the automatic decoration on the server side, e.g. for Wordpress at http://sw-guide.de/wordpress/link-indication-plugin/.

minimalism: link popups are a compromise of providing functionality only on demand, and providing it as soon as possible.

The standard interaction technique used could be improved by increasing sensitivity towards context. The information structure delivered by Hyperscout thus must trade off consistency in placement, allowing spatial memory to support the digestion of the presented information, and conciseness, suppressing information that is not useful within the context. For some applications (e.g. search engine results), the presentation of information within tabular lists has advantages over popup information.

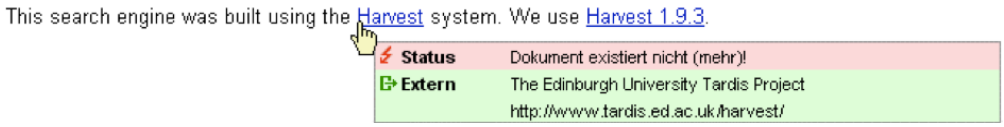


Figure 61: Hyperscout popup showing information about a dead link.

Using popups or tooltips for the display of meta-information has also the handicap that different links cannot be compared; Hyperscout was extended to use „thumbtacks" to this end (Witt, & Tyerman, 2002). However, this places further demands on the user, and the question remains whether it would not be easier to compare the target pages directly. For search engine results, the information for different links will often need to be compared. Annotations using popups (used e.g. in Sharon, Lieberman, & Selker, 2003) do not allow the side-by-side comparison. Direct changes of the link marker visualization, e.g. coloring links according to implicit types (Obendorf, & Weinreich, 2003), can be useful for providing information without creating the need for a user to use popups (Weinreich, Obendorf, & Lamersdorf, 2004).

Discussion

The Hyperscout case demonstrates that not only the structure of information, but also its visualization can affect the resulting access structure. Hyperscout provides a minimal interface as it does not add visible information to Web pages at first. Information about target pages is provided only on demand. The structural minimalism in this is that the original interface—a Web page filled with text, graphics and hyperlinks—remains virtually unchanged. Only when it becomes necessary for the user to decide whether to follow a link or not, decision support is provided.

This marks both a strength and a weakness of the concept. The user can freely decide whether he wants to trade off speed and conciseness for completeness and certainty. If response and rendering times continue to drop significantly, the temporal cost difference between displaying the result page, and information about the result page is reduced. As meta-information can be incomplete and erroneous, this will shift the balance to the disadvantage of Hyperscout. A further problem is that although the look of the interface in unchanged using Hyperscout, the feel changes: informal observations indicated that usually, users don't hover over links before they click on them. If users liked Hyperscout, their behavior changed, and an additional navigation step (read–hover–click) was introduced.

As users tend to rapidly navigate on the Web—in a recent study, we found that most pages are visited only briefly, with 25% of all documents displayed for less than 4s and 52% of all visits being shorter than 10 seconds (Weinreich, Obendorf, Herder, & Mayer, 2006b; Weinreich, Obendorf, Herder, & Mayer, 2006a)—this additional step could slow down navigation behavior. Yet, while we previously found that small changes, such as the marking method for links could have large effects on reading behavior (Obendorf, & Weinreich, 2003), quantitative data is difficult to interpret as many different activities are overlaid in the data (Obendorf, Weinreich, Herder, & Mayer, submitted; Herder, Weinreich, Obendorf, & Mayer, 2006; Weinreich, Obendorf, Herder, & Mayer, 2006b). Also, if the information was really helpful for the user, those navigational steps that only deal with retracing detours would become obsolete.

Following a user study that tried to study what information would be most useful for the users, version 2 of Hyperscout was developed. Here, interviews examined which information would be useful for each link type (comp. Weinreich, Obendorf, & Lamersdorf, 2004). The resulting version displayed much less information, concentrating on the most useful items in the pop-ups. Further options were added that marked out e.g. external links on demand.

Table 10: Minimal Aspects highlighted in Hyperscout

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: While the Hyperscout system is technically complex, with filter, crawl and proxy mechanisms working together, backed up by a database, the functionality is simple: displaying additional information as the user moves the mouse over a link. In the second version, first experiments were made what other uses the existing information could have.

Structural Minimalism: As Hyperscout competes with Web download times, and rapid navigation events, the most important requirement that users named during the test runs were the immediate accessibility of information „at one glance". A categorization of different link types, and of automatically generated meta-information allowed the prioritization of pertinent information that is displayed for different types of links. Hyperscout thus does not base its minimal structure on frequency of use, but on the related measure „perceived usefulness" for link types.

Architectural Minimalism: Hyperscout is a tool adding functionality to Web browsers. Although it is technically realized as a Web intermediary, this is only due to its prototypical status; a final release would rather be integrated into the browser. It would still add features almost without complicating the interface as its functionality remains invisible unless explicitly invoked by mouse-overs, or modifier keys.

Compositional Minimalism: Due to its specificity, not only in the information provided, but also in the structure of information, and the mode of interaction, appropriation of Hyperscout is unlikely and has not been observed.

5.2.5 Word Processing

Word processing applications have acquired functionality that extends far beyond the task of processing words alone, they are able to manage bibliographies, print serial letters, and produce sophisticated layouts. The number of features in Microsoft Word has risen steadily. This is not only a common perception, Microsoft itself has documented the feature increase in an official Blog on the usability of Office 2007 (Figure 57, Harris, 2005).

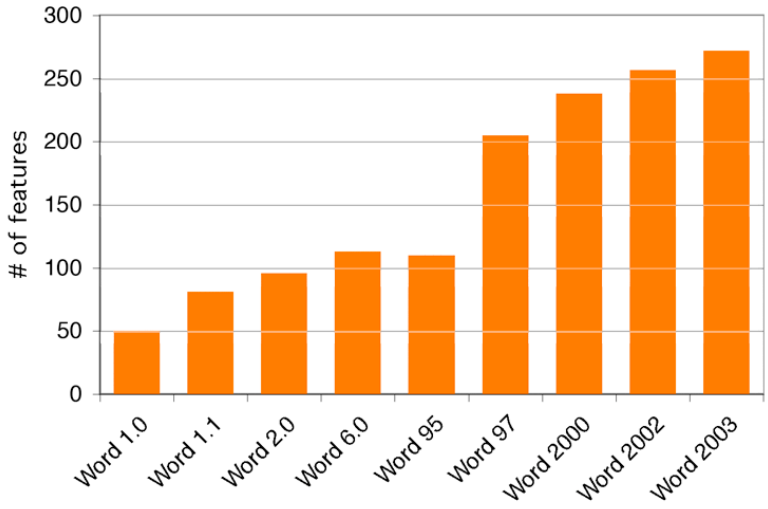


Figure 62: Features of the Word product line. (Harris, 2005)

The accumulated features have created the need for an increasingly complex interface. Judging by the number of tool bars and task panes alone, the rise in interface complexity is even more dramatic than the accumulated functionality (Figure 58, ibid.).

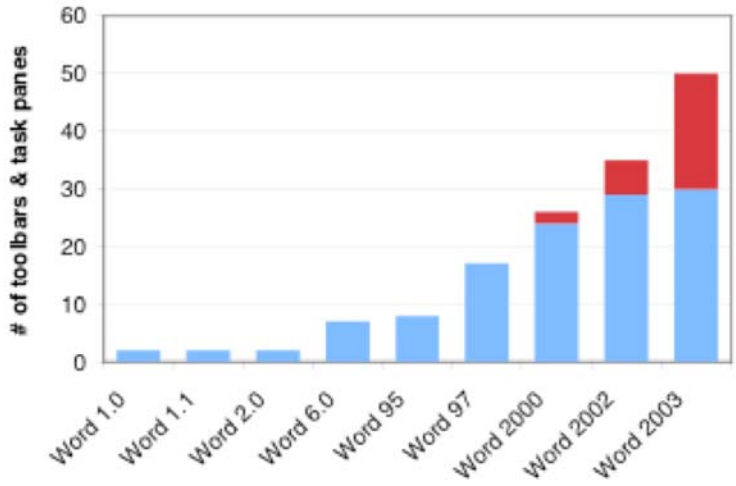


Figure 63: Number of toolbars and task panes in Word applications. (Harris, 2005)

While word processors have become very complex applications, the complexity of work tasks is unlikely to have increased proportionally. This means that many users use only

few functions. The use of functionality follows a Zipf distribution: few functions are used very often, while others are used much more infrequently. However, most functions are used by someone, the resulting distribution of used events is a long-tailed distribution (Harris, 2006f). The five most often used commands for Word 2003 were paste, save, copy, undo and bold. Paste actions alone account for more than 11% of all commands, and together these five actions are performed in 32% of all cases (ibid.). As the designers of Office believe that all functions are used by someone, removing functions was not an option—the intended goal was rather to make users use more features (Harris, 2006a). The question was thus how to adapt the access structure in a manner that would provide easy access to often used functionality.

Adapting structure to fit the users' needs has long been a focus of research, and there exist many techniques to design menu systems according to users' expectations and needs (Norman, 1991; Obendorf, 2003). It is also not a new idea that systems should ‚behave intelligently' by adapting their access structure automatically to better meet the individual user's needs. Various approaches for adaptive menus have been discussed in the literature (e.g.Mitchell & Shneiderman, 1989; Sears & Shneiderman, 1994), and although some experiments showed that techniques like split menus can decrease task time, other experiments showed that simple ordering by frequency did not improve efficiency of users—while it might generally be assumed that the mild disorientation caused by changing menu layouts could harm user satisfaction. A recent controlled study found that users both prefer adaptable menus to adaptive menus and work more efficiently with the former (Findlater, & McGrenere, 2004).

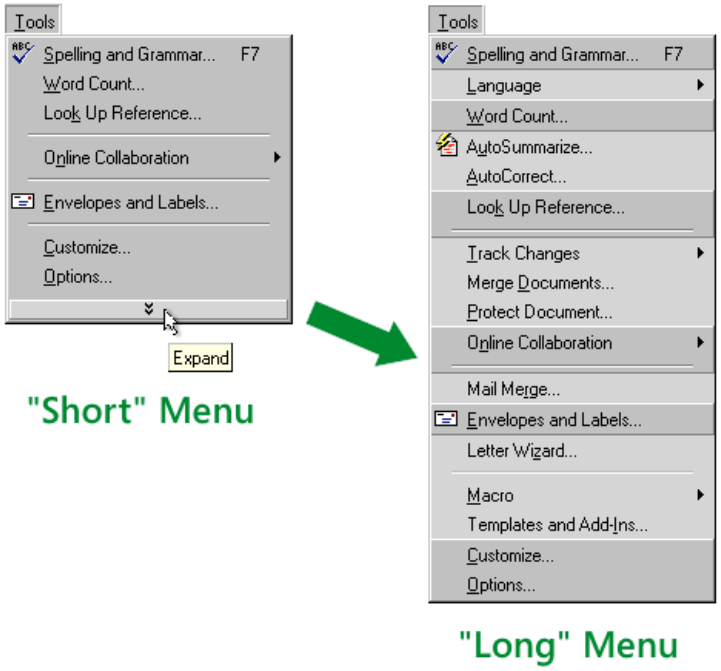


Figure 64: Adaptive menus in Microsoft Word 2000: after a click on expand, the full menu is shown.

Nonetheless, for its *Office 2000*, Microsoft chose to implement adaptive menus (Figure 64): „Word 2000's new adaptive menus display only the most commonly used commands at first, and then expand to show all of their commands" (Rubin, 1999, 163). Infrequently used menu entries were first hidden, then grayed out when the full menu was requested. However, in contrast with the menu system of e.g. the Palm, Microsoft's solution had two drawbacks: the absolute position of menu items would change as other items were included or excluded automatically, and a menu item that was important but would be used infrequently could disappear. This behavior of the menu system irritated many users as they could not understand why adaptation was taking place; even if they understood the underlying mechanism, it was working against them and forced them to wait for the expanded menus to select their target item. As menu recall is aided by spatial memory (Norman, 1991, ch.13), the adaptive menus were prone hurt the users' feelings, but also the measured task performance.

Microsoft also implemented adaptivity for toolbars. The so-called „rafted" toolbars were displayed when not all icons would fit on the screen. Some icons were moved to a drop-down dialog that would appear upon clicking on an expand arrow. The selection of icons in these rafted toolbars also depended on the frequency of their use: often used icons were moved from the dialog to the bar, where infrequently used icons had to make room. Thus, Word 2000 was trying to provide an exact fit to the user's use frequency, trading off consistency of placement.

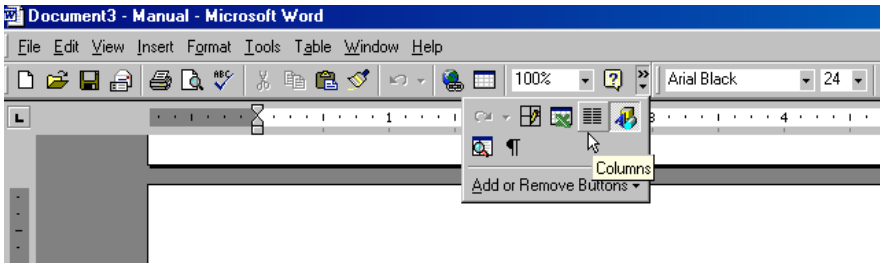


Figure 65: A ‚rafted toolbar' in Microsoft Word 2000.

Description

While adaptive menus create a minimal access structure if one takes frequency of use as the primary measure, the question remains whether the resulting menus form a whole—no vital options are left out—and whether frequency of use is a good indicator for personal importance. Problems with adaptivity have been listed in a review of the current state of research by Christina Höök (2000) as adaptive systems may create a *lack of control* for the user—mechanisms for adaptation are often complex, or cannot controlled at all. They can also cause *unpredictable results*, decrease *transparency*, intrude the user's *privacy* and reduce the user's *trust* in the system.

Some of these problems also surface in practice: users seem to generally dislike adaptive menus in Microsoft Word 2000, and among a list of the „top 12 tips for MS word" users by the author of *Word 2000 in a Nutshell* (Glenn, 2000b), number 1 reads: „Turn Off Adaptive Menus" (Glenn, 2000a). Personal preference of users was found by Findlater

and McGrenere to tend towards manually adaptable menus in a controlled study (Findlater, & McGrenere, 2004), and in an experiment with different interfaces for Microsoft Word, McGrenere also found a preference for individual tailoring (McGrenere, Baecker, & Booth, 2002). After Office 2003, Microsoft seems to intend to switch off adaptive menus by default (Harris, 2006b).

With the newest version of its Office suite, Microsoft is going to introduce a number of dramatic changes to the interface. „The user-interface is startlingly different – and better" according to Darren Strange, Product Manager for Office in the UK (Hales, 2006): „In the first Word for Windows there were 100 features … while in Office 2003 there were 1500, all buried in the program in places many users dare not go … users found it difficult to remember where everything is … [so now] everything will be contextualised."

Figure 66: The ‚ribbon' provides access to contextualized functionality; ‚tabs' switch between contexts.

In the new interface, a ‚ribbon' replaces all menus. It thus unifies access to features that were previously accessible either by menu or using a toolbar. Instead of the different pull-down menus, the ribbon provides several ‚tabs' that group functionality. While the menus, however, grouped functionality according to consistency with other applications, and according to objects and attributes that could be manipulates (format, table), and had little success finding a better label for those features hidden in the tools and edit menu, the ribbon groups Word's increasing number of features in tabs that relate to steps in the work process.

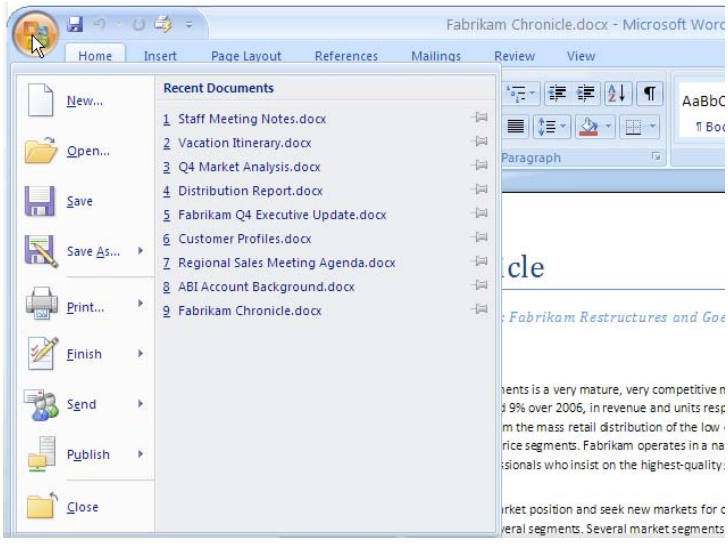


Figure 67: The ‚Microsoft Office Button' provides access to most often used functions.

On the other hand, an increasing adaption to certain tasks bears the risk of too narrowly defining the user's course of action, and limiting his freedom to choose the right path of action. This aspect is further investigated below (see *compositional minimalism*), but it has also been addressed by Microsoft: with the introduction of a generally accessible menu containing the most important options (called „Microsoft Office Button" in the public beta version), and with the ability to quickly change contexts using tabs, the user can both access basic functionality at any time, and choose between access structures tailored to a certain context.

While it was possible in previous versions of Office to show e.g. image manipulation toolbars only when an image was selected, this has now become the default behavior. The number of active toolbars is thus massively decreased—at the cost of having an interface that changes according to the current selection: as the user selects e.g. a picture, an additional tab becomes available (Figure 68).

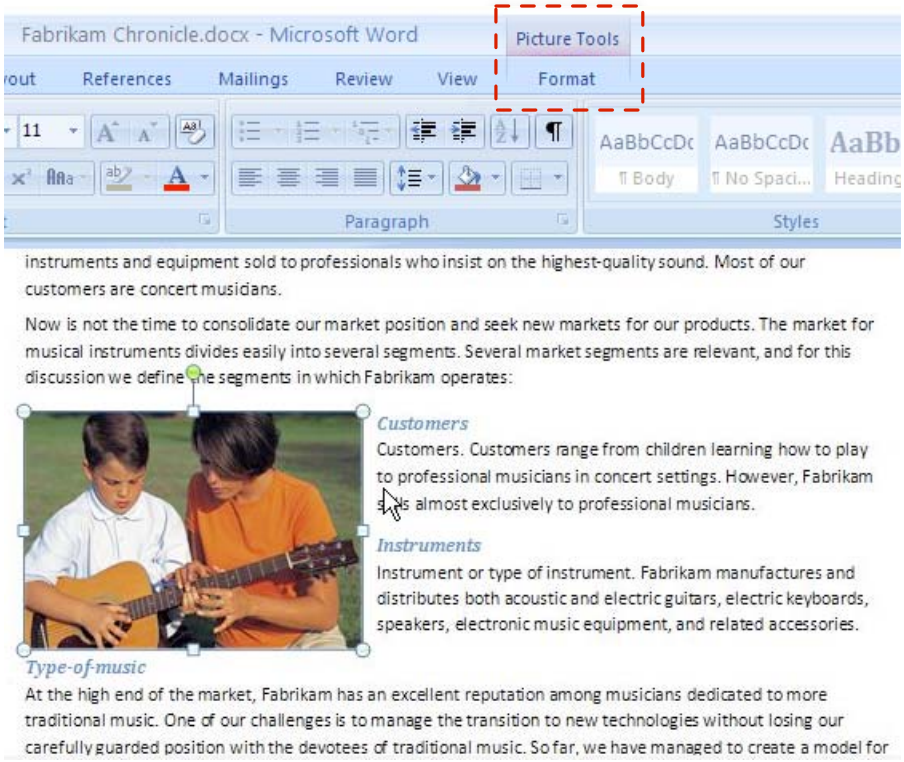


Figure 68: Contextual tabs only appear when certain content types are selected.

Discussion

The underlying principle of Microsoft's approach in Office 2007 is contextualization; the complex functionality of Word is broken down according to use contexts—Word still appears as a single, powerful tool, albeit with contextualized menus. This approach has been used very successfully in context menus (comp. Koved & Scneiderman, 1986) appearing on a right mouse click. The difference is here that the new ‚ribbon' interface makes the contextualization immediately visible; in this respect, it bears some resemblance to in-

spector windows that appear only for certain types of selections—this approach has long been used e.g. in Powerpoint for processing images. Comparable functionality existed in previous versions of Word as Toolbars would only be displayed as necessary. The difference, however, is that the ribbon adds an additional, optional *mode* without adding to the interface complexity whereas appearing toolbars immediately complicated the interface. As an alternative approach, *architectural minimalism* is discussed in the next section; in contrast to the contextualized interface presented here, it focuses on presenting several distinct tools that can be used together to provide the designed functionality.

5.2.6 Refining the Notion of Structural Minimalism

Structural minimalism was introduced as the question how to order functionality in a manner that minimizes the perceived structure for the user. Remote controls as examples (1) introduced the mapping problem, and demonstrated the shift to other modalities. The design of the Palm and the Navi-Key demonstrated (2) how usability tests can generate an optimal fit of the access structure to sub-tasks and (3) how frequency-based selection of functionality can generate the impression of simplicity. Hyperscout posed the question when to (4) actively hide functionality, and the redesign of Word 2007 suggests to (5) adapt the access structure to activities rather than to the use frequency of functions.

It is unclear when reducing the number of physical controls increases the usability of a design. Contradicting the conclusions from the remote control example, Norman (Norman, 1989) favors his car's interface over his telephone's: Automotive engineers and designers have traditionally mapped new functionality to new controls, increasing the number of controls. With minor exceptions, there is one control for each function, and each control can be labeled naturally. By contrast, in Norman's reference phone, 24 functions map to only 15 controls, none of which are labeled for specific actions. According to Norman, the visible car controls remind the user from the available possibilities, unlike the phone with unlabeled controls telling nothing about the device functionality. The good relationship between the car controls and what they do also makes it easier for the user to master the car's function—especially for fast moving vehicles, it can be dangerous to reduce the number of controls, and shift the access structure into menu systems.

This benefit was lost when BMW introduced its iDrive system. The habitual mapping between controls and their functionality was lost as the iDrive introduced a turnable knob controlling a complex menu system—a single control for all of the car's advanced functionality. Jef Raskin attributes this to the introduction of modes: „Turning the iDrive knob shouldn't mean different things in different modes. You shouldn't need to stop and ask, 'What mode is this thing in right now?'" (Wilkinson, 2002) However, there seems to be a limit to direct mapping—the 112 controls in Don Norman's Mercedes Benz fail to compare to the 2002 BMW 700 series with its controversial iDrive interface: One of its designers noted: „The people who designed the interface, we didn't need 700 functions. We always discussed whether we need this function or that function, because it would

have made it for us much easier to build a simpler system. But OK, if our marketing department says we need it, we design it in" (quoted in Kiljander, 2004, 159).

The design of the Nokia 3300 series highlights that it can be useful to differentiate between the *actual* and the *perceived ease of use*. Although the Nokia designers do not give a definition for measuring the actual ease of use, and problems, such as whether a more sophisticated interface can be used more efficiently after training, or whether a direct mapping between interface and functionality is desirable, remain, making this distinction allows to extend design beyond the measurable usability, and towards the users' experience: reducing an interface visually has an important effect on users, adding to their perception of simplicity.

However, if the overall complexity cannot be reduced, the trade-off introduced by making a system structurally minimal can result in less usable systems. John Maeda introduced the *Goldilocks* metaphor for hiding the right amount of supportive information in the forest of consumable information: navigation in a beautiful forest motivates the provision of not too large signposts—they would defer attention from what is really interesting (Maeda, 2005b).

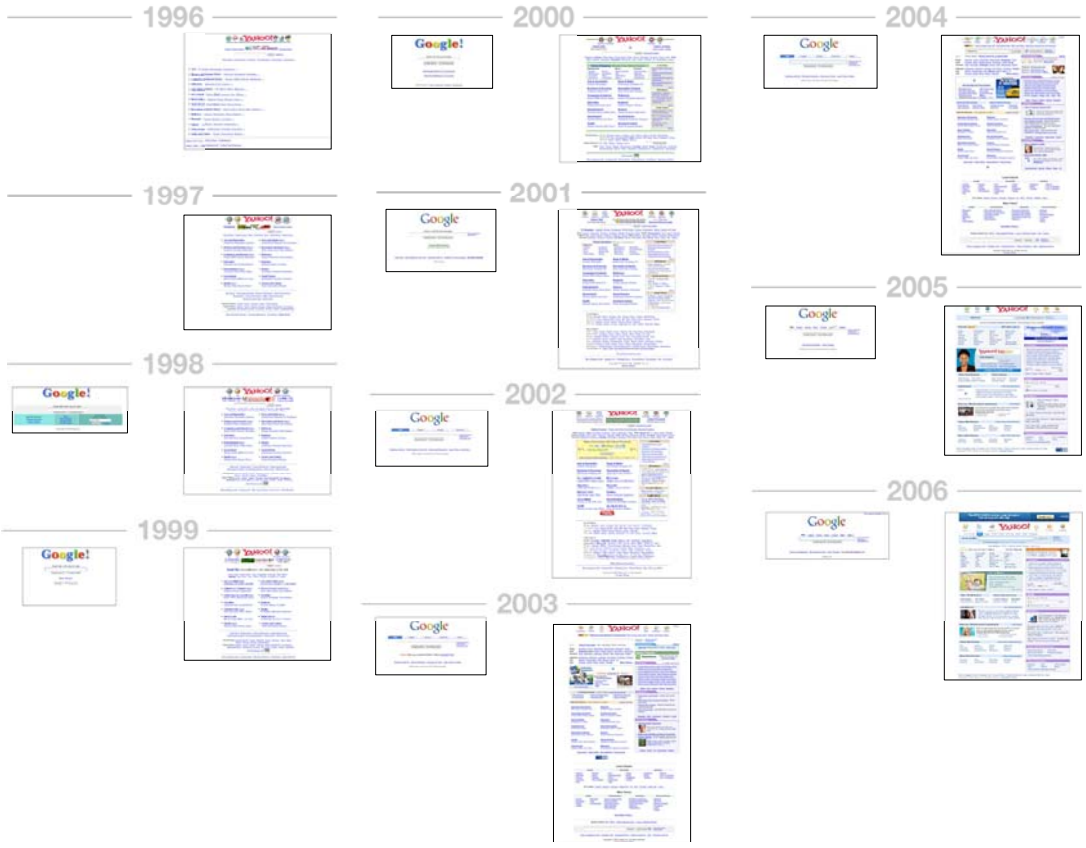


Figure 69: Yahoo and Google home pages from 1996 to 2006.

The question whether functionality should be actively kept hidden is not only crucial for research projects, Google as one of the major Web companies faces similar problems: its

home page is commonly rated as very simple—and as very easy to use. It features most prominently a search form, and a submit button. If one compares this to Yahoo, Web.de, or other large portals, the Google home page indeed seems to be much simpler—although it has become somewhat more complex over time (compare Figure 69). However, its simplicity is owed largely to obscurity: many of the additional services that Google offers (Google Maps, Google Mail, Google Earth etc.) are never mentioned on its home page. This has led Don Norman to comment: „is Google simple? No it is deceptive!" (Norman, 2004b) More importantly, Google realized it hurt itself with its strategy when they launched Google Video, and movie sales were hurt by the fact it was not mentioned on their home page (Post-Intelligencer, 2006).

The Nokia example raises another issue for structural minimalism: is „intelligent" (read automatic) adaptation minimal? By definition, minimal structures are those that are not visible to the user. If a design adopts its structure to the task—as e.g. Microsoft Word 2000 does with its toolbars and adaptive menus—its risks to disappoint the user by failing to meet her expectations. It seems that this problem is fairly common, as many adaptive systems are only designed with an incrementally optimal fit to task in mind, and disregard effects of the observable changes in access structure. While this is not sufficient evidence to deduce that adaptive systems cannot be structurally minimal, the examples for minimal structures found here are static: they focus on *reduction by design*, rather then *designed reduction*.

The application of *structural minimalism* to the Palm design, but even more to word processing software, demonstrated a central difference of structural and architectural minimalism: in the case of structural minimalism, the complex functionality of a single tool is organized according to tasks. While the new Office 2007 will probably introduce a few hundred new features, it tries to group access to this abundance of functionality in accordance with the users' needs. Where information is given preference to function, this process is also known as *information architecture* (comp. Wurman, & Bradford, 1996; Rosenfeld, & Morville, 2002). While this introduces a terminological overlap with *architectural minimalism*, the meanings of these terms differs decisively: information architecture is often used as an umbrella term for the graphical and structural design of Web sites (ibid.), or takes on an even broader meaning, including questions of emotion or empowerment (Dillon in Carliner et al., 2001). Both structural and architectural minimalism can thus be seen as focused perspectives on information architecture, only that their design scope extends beyond Web pages.

5.3 Architectural Minimalism

The whole is simpler than the sum of its parts.
—Josiah Willard Gibbs[29]

The notions of *functional minimalism* and *structural minimalism* concentrate on properties of the interface—what functions are provided, and how access to functionality is adapted to a task. *Architectural minimalism* denotes a minimality in the constituents of an interface, and assumes that the interface itself can be modularized. The „interface" is thus expected to consist of several visibly distinct tools, interaction is split into distinct dimensions. As examples for architectural minimalism are described here based on observations of interfaces, but the notion is defined as resulting from the underlying architecture, this analysis serves as an indicator whether the distinction of functional, structural and architectural minimalism is useful in practice.

5.3.1 Building blocks

The 1958 Lego brick is an example for the possible complexity that can be created by combinations of very simple building blocks. Lego bricks have been used to model lifelike scenes, or recreate hallmarks of civilization. There are two lessons for architecture built into the bricks: First and most in the spirit of the minimal, identical blocks can be combined to form most complex shapes (Figure 70b). Many Lego builders make exclusive use of the ‚original' Lego brick[30] (Figure 70a); the architecture of their models is based on the primitive cuboid.

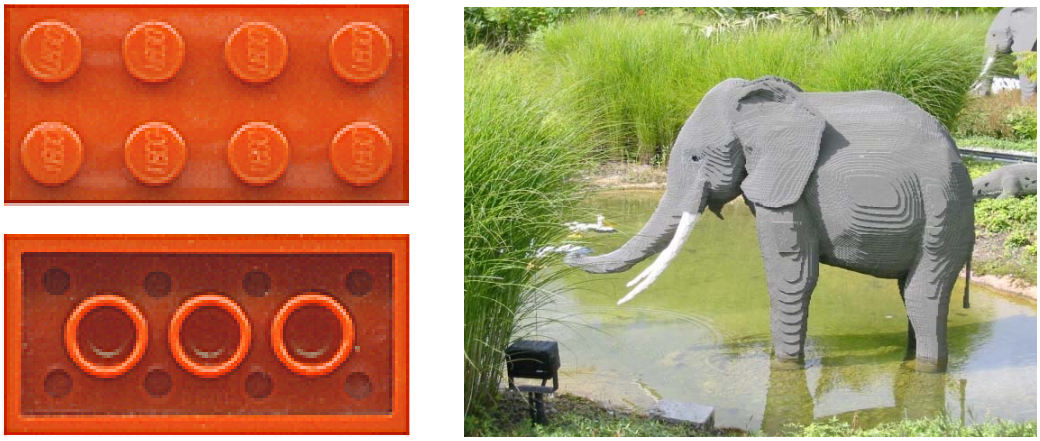


Figure 70: The original Lego brick, and an animal made of some of its brethren. lego elephant

29. Gibbs, a physicist and mathematician, lived from 1839 until 1903 and worked as a professor in Yale. The citation cannot be reproduced here without alluding its original context, thermodynamics (cf. Jaynes, 1992).

30. The first bricks were copycats of a British invention, the *Kiddicraft Self-Locking Building Brick* from 1940, and were introduced in 1949 under the label *Automatic Binding Bricks*. They lacked the *Stud-and-Tube Coupling System* that holds together today's bricks; the tubes on the underside increase the stability of the bricks, and, most importantly, the grip on other bricks. Only thus, complex constructions were made possible and the product became a huge commercial success. (Lithgow, 1987; Hughes, 2005)

Although the backwards compatibility of Lego bricks is unequalled in computer science—bricks fresh from the store mix nicely with those inherited from the previous generation—some changes have occured. The second lesson of the Lego brick is that although new forms are continuously introduced (more than 2500 different LEGO elements have been designed[31], the binding principle remained simple, and thus the ‚original' brick still interfaces with all new elements (see Figure 72).

The artful composition of standard, mass-production tiles can create an infinite wealth of different forms. Using only the rectangular basic Lego brick, sophisticated models have been formed by Lego enthusiasts all over the world, and the Lego amusement parks rebuild real landscapes, buildings, or lifeforms by individually combining bricks to form new shapes. Sometimes, special pieces are used that have been designed by Lego for that purpose, but most model builder base their designs on the original 4 x 2 brick.

Figure 71: New Lego elements.

While the standard Lego sets are still forming the main business, Lego had some success introducing the Lego Technic series where simple mechanics can be rebuilt using special tiles. An important success factor was the compatibility to standard tiles, enabling model builder to integrate mechanics into their designs. Lego also built the Mindstorm robots, whose first version did not create a universal commercial success—the robot series became immensely popular for teaching and research purposes (Klassner, 2002; Hood, & Hood, 2005), but many end users felt the included software placed unnecessary restrictions on the possible designs, and few customers were able to custom-build their own driver or operating system for the robots[32]. Lego manager Lund believed that „Mindstorms' main flaw … was its complexity; many kids lost interest before completing their first robot … Lund wanted novices to be able to construct and program a robot in 20

31. http://www.lego.com/eng/create/designschool/lesson.asp?id=1_a

32. Examples for user-supplied drivers can be found at http://www.crynwr.com/lego-robotics/ and the operating system BrickOS is available at http://sourceforge.net/projects/brickos/.

minutes. The biggest barrier to making that happen was the Mindstorms programming language, known as RCX-code. Though simple by computer science standards, it was too frustrating for many programming neophytes". (Koerner, 2006) For its new robot series, Lego partnered with Microsoft (Microsoft, 2006c) to provide both a novice, and an expert programming environment. Although Lego discontinues packaging the classic tiles in the new NXT Mindstorms series as they want to change to a more modern look, the studless Technic tiles are still compatible (Koerner, 2006).

Discussion

The example of Lego bricks highlights a necessary differentiation to understand ‚architectural' minimalism correctly. The notion of *architectural minimalism* refers to the architecture of the interface, providing different tools that can be recombined by the user to adapt the designed system to her needs. In the context of information systems, architecture is often understood as an internal aspect of a design that is not necessarily visible to the user.

The building blocks of Lego could be set in relation to the plugin concept that can be found e.g. in the Eclipse development environment. However, this assumes that the user is the spectator, not the builder: functionality can be configured by adding or removing plugins, yet these do not necessarily show up in the interface, and if they do, they are often not recognizable as individual tools, but instead aim for a close integration with the existing interface, *only adding* buttons, menu items and automated behaviors. If plugins were designed so that end users could adapt them, they would rather follow an *architecturally minimal* approach to interaction design, differentiation and combination would be valued over integration—the additional functionality would only become active as the user adapted the tool for his needs. Although Eclipse is often presented as an example for exemplary user-interface design, and although it certainly introduced many excellent ideas that were adopted by other integrated development environments, it should be kept in mind that the prototypical Eclipse user is a programmer, and thus both in the role of configurator and user—an important difference to typical end users. By contrast, the visibility of the building blocks played a decisive role for the Mindstorm robots: the lack of transparency that users perceived in the first version of the Lego robots could partly derive from the failure to provide tools for individually programming the individually visible physical parts.

The power of user-controlled adaptation is clearly demonstrated by the Lego bricks: as users build models, they ‚interpret' the simple rules provided by the Lego system, and their interpretation becomes more important than the design of the individual brick. The Lego brick demonstrates the strong force of internal consistency and interoperability of tools: complex designs can be created with simple building blocks as they interlock and build upon another. The design of more and more specialized bricks on the one hand gives these creative builders more expressive power, yet on the other hand, it generates additional sales and broadens the user base to also include those who are satisfied with recreating pre-designed models.

5.3.2 Apple Automator (i-Series 2)

When Apple designed the i-Series applications, their main intent was to provide the *core functionality* that most users would want to use. One answer to the consequently rising expectations in terms of functionality—as users learned to use the first, simple version of a tool, they discovered new requirements—was the gentle introduction of new features, with a careful eye on the wholeness of the design (compare 5.1.2).

Description

Another approach that targets „power users" who wish to use new functionality that will only be needed by specific groups of users is exemplified by Apple Automator: Almost every Apple application designed for consumers (Mail, iCal, Address Book, iPhoto, iWeb, GarageBand, etc.) exposes an application programming interface (API) that can be used to script the application using the AppleScript programming language. Part of this functionality can also be used by the Automator application. Automator allows users to graphically design custom workflows. Users can e.g. process files in specific directories, create image filters, auto-mail processed content to specific addresses, download and store news papers, or convert text to PDF files—repeated tasks that would normally be taken care of by script programming. Automator promises to enable end users to create such scripts, or „workflows" in its own terminology.

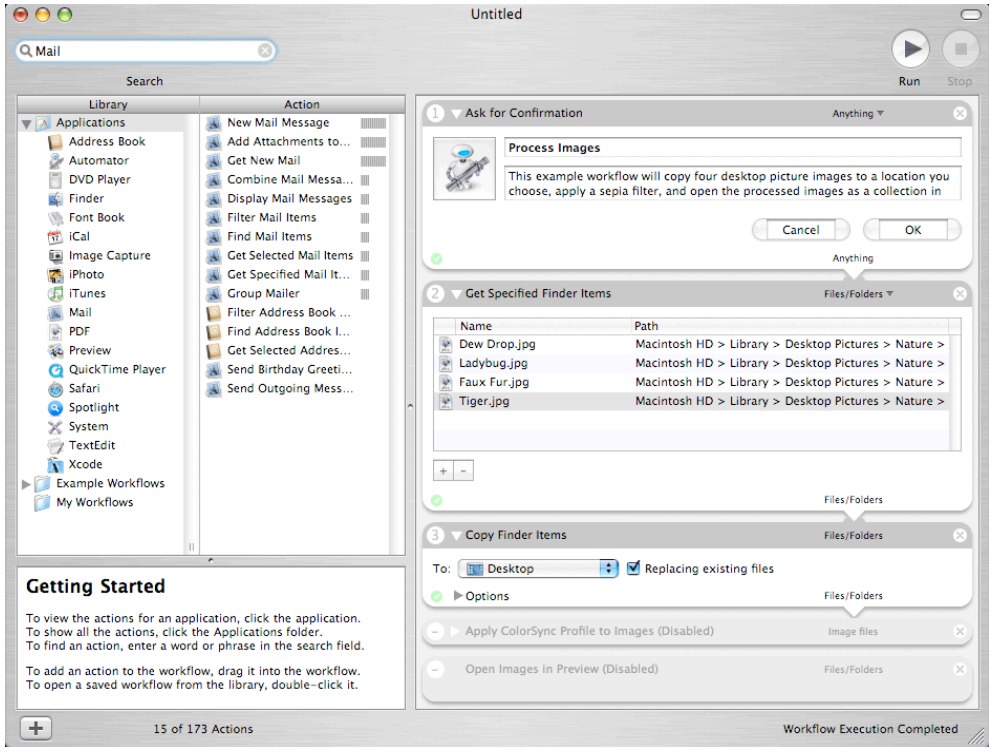


Figure 72: The interface of Automator 1.0 with a script on the right, and actions listed on the left.

Automator features a graphical interface instead of a textual command language, and guides the creation of a workflow by allowing only correct types of information to flow into the next action. Individual actions are first chosen by selecting the application and

the desired functionality, and dragging it to the workflow area on the right. Options for actions can be modified by parameters changing the result of the action, e.g. providing storage locations, or selecting a compression quality. Apple has also consciously chosen not to implement control structures—there are no conditionals or loops available (comp. Rosenthal, 2006).

The combination of several applications is often capable of delivering the functionality that would require a single application to be much more complex. Apple follows this line of thought and exposes the functionality of its entry level applications to the Automator application, which tries to enable end users to create their own customized features. Users, however, cannot generate new tools. Rather, actions within tools that are already used in conjunction by a user manually are automatically combined, and a new single-purpose function is created. For the Apple i-Series, the Automator concept eliminates the need to cater for all use cases; responsibility for programming is redistributed to the user who can build a specific work environment for himself.

Table 11: Minimal Aspects highlighted in the Apple i-Series (Automator)

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: The Automator tool has only a specific function: drawing together the functionality of other applications that form a part of the bundles that come either as part of the operating software, or in form of inexpensive suites for multimedia (iLife) or office work (iWork). Each of these applications has only a very specific use in itself. This creates both the need for additional functionality, and the possibility to create part of this functionality through new ‚automated' combinations between tools.

Structural Minimalism: While the Automator builds upon the simplicity of access to the tools' services, the resulting access structure is the creation of the end-user programmer. As the designer has no part of it, and no technique is employed to ensure structural minimalism, it is of no importance to the discussion beyond the access structure of the Automator tool itself; some reduction of the high complexity of end-user programming is created by dispensing with control structures, and through typing of the information piped through the tools' services.

Architectural Minimalism: The division of responsibility for work tasks between different applications *forms the basis* for Automator—without this architecturally minimal system design, scripting would only be possible within single, monolithic application. The standardized API exposed by the installed applications allows Automator to use the applications as *interoperable tools*. As users are already accustomed to switch between tools and use several different tools on a single work task, the aptly named Automator only *automates* their tool use, thereby creating a new, specialized workflow. The new workflow tool draws on services the individual tools provide, but is specific to

a given task; strictly speaking, it does thus not provide additional functionality, but rather a shortcut to access existing functionality.

Compositional Minimalism: End-user programming is *one* approach that allows the user to determine the use of a tool—instead of offering choice between processes, individual services are provided that the user, or a supporting expert, can tailor to suit a specific use situation. While the existing applications in Apple's i-Series provide ready-made bundles of these services, using Automator users can create new sequential bundles of functionality, and distribute these as scripts to other users who either make direct use of them, or base their specific customizations on them. From the perspective of *compositional minimalism*, however, this corresponds more closely to a second-level *composition* than to an *interpretation-in-use*. This composition of effective and efficient workflows complements the freedom of use when users employ another tool through the many pre-defined links already implemented in applications (e.g. use mail for sending email, iCal to store appointments, Address Book for contact information).

5.3.3 SketchUp

Sketchpad by Ivan Sutherland revolutionized both computer graphics and human-computer interaction in the early 1960s (Sutherland, 1963); it was the first interactive object-oriented direct-manipulation drawing application—in a time where an average computer took the space of a whole room for itself and input was still mostly done by punched cards, it featured an interactive graphical display with a light pen.

Since computer-aided design (CAD) arose as a discipline in the 80s, most software required the user to draw polyhedral surfaces in wire-frame (Lipson & Shpitalni, 1996), or input exact data using numerical dialogs. 2D input for 3D construction has always been cumbersome, and more complicated than simple drawing, so simpler approaches was sought. Researchers have tried to target unintuitive input by using drawing techniques that allow unambiguous interpretation in three dimensions (Eggli, Brüderlin, & Elber, 1995). Systems like SKETCH and Teddy demonstrated that gesture-based interfaces are a powerful and intuitive base for 3D model design (Zeleznik, Herndon, & Hughes, 1996; Igarashi, Matsuoka, & Tanaka, 1999). However, these approaches need to trade their simplicity against limitations on the appearance, or on the topology of the generated models. SketchUp[33] is a commercial product targeted at the early phase of designing interior and exterior architecture, and builds upon some of the techniques listed; in contrast to the research prototypes that experiment with gestures, SketchUp's qualities lie in the excellent facilities for combining different tools.

33. I have been using Sketchup as an example for minimalism since around 2003; apparently, Google has also found the application noteworthy and bought it in 2006 (SketchUp, 2006)—now, there's a limited free version called Google Sketchup, http://sketchup.google.com.

Description

SketchUp presents the user with a conventional 2½-D projection window. The interesting part of SketchUp is the minimal number of tools available to the user, and the ability to combine them. As the name indicates, SketchUp is largely based on sketching techniques; the main tool for object creation is a pencil. As a 2½-D projection would normally present many ambiguities for positioning the pencil cursor in 3-D space, the pencil tool and many other tools feature automatic alignment, e.g. to planes, or existing surfaces or edges. This makes it extremely easy to sketch 2-D surfaces in the perspective view. Alignment can be controlled as existing vertices, edges or faces can be selected as cursor guides; e.g. a line can easily be made the same length or orientation as another visible line, or an existing face can be used to find its midpoint and use that as a guide[34]. Closure is automatically detected by the program, creating faces from closed shapes.

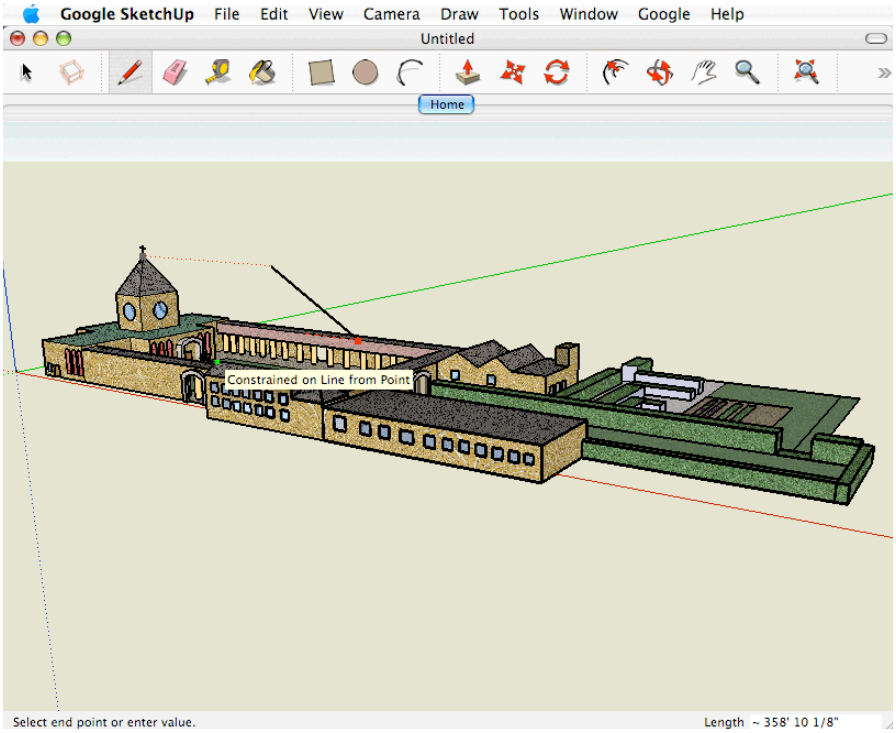


Figure 73: Google SketchUp.

The creation of three-dimensional bodies is also consequently controlled by tools in SketchUp—there is no need to use extrusion commands as is common in other construction packages. Instead of these number-based dialog interactions, a push/pull-tool can be used to extend a selected face into three-dimensional space. While at first this allows only perpendicular surfaces to be generated, edges can easily be moved afterwards to generate more complex forms (Figure 73).

34. It is inherently difficult to write about dynamic interaction. The 1:08 hour-long demonstration video provided by SketchUp (http://download.sketchup.com/downloads/markets/SketchUp_Live_demo.wmv) does a much better job of illustrating the rich interaction made possible.

With only these two tools and the alignment features that remind of Sketchpad, complex three-dimensional bodies can be constructed. A review stated, „SketchUp has a minimalist easy-to-learn interface … SketchUp's simplicity does not come merely from a stripped-out set of 3-D modeling features but from the development of some very innovative ideas of how 3-D modeling can be made easier and more intuitive. The application actually incorporates some very complex inference algorithms that constantly look for alignments, snapping, and directions based on the mouse movements, minimizing the input that is needed from the user to perform modeling operations" (Khemlani, 2004)

Other tools can be used as necessary, there is a tool for curves, for circles, for rectangles, and for freehand drawing. Furthermore, there are tools to manipulate the geometric models created with the drawing tools, such as tools for color or texture manipulation. Since the application has been bought by Google, tools for obtaining and sharing models, and for integration with Google Earth have been added.

Discussion

Although SketchUp's application domain, the construction of three-dimensional models, is one of the most sophisticated uses for computers, SketchUp succeeded in providing an unusually easy-to use interface: „The simplicity of … SketchUp makes three-dimensional drawing possible even for those who lack knowledge of design or perspective." (Glinert, 2006) SketchUp does not aspire to replace modelers targeting photorealistic rendering, but it is widely used as a sketching tool for architecture, and also for interior design (Lok, 2004).

There are several aspects of SketchUp which can be considered minimal. It is not functionally minimal, although it lacks some of the sophisticated—and complicated drawing functions of other software. It provides a multitude of functions distributed over dedicated tools that work intensely hand-in-hand. This alone might qualify SketchUp as an example for architectural minimalism—but it does not distinguish the application from other object-oriented drawing applications.

What is really unique about the way sketches are created with SketchUp is the complex functionality hidden within the automatic alignment function, and the interaction of drawing tools and the push/pull tool that manipulates a different dimension. The latter is more easily illustrated without hands-on experience: Sketching in SketchUp is often first done in a planar fashion. This 2-D sketch is then used as a basis for creating a 3-D model; e.g. starting from an architectural outline to model buildings of different height, two dimensional faces are „pulled out" or „pushed in" to interactively create a three-dimensional body. While this limits the modeling of the third dimension at first to perpendicular movements, faces can later be moved to create skewed bodies. While the traditional extrusion mechanism used in modeling applications allows a very precise specification of dimensions, it does not create the immediate effect of Sketch-Ups push/pull tool.

The inaccuracy of the direct-manipulation modeling is countered by alignment interaction techniques: all elements of the existing sketch can function as guides. While automatic perpendicular guide-lines are created from all surfaces and edges that are aligned

with the currently pushed/pulled face—and within reasonable proximity, and from their midpoints, specific guides can be used by depressing a modifier key whilst over a certain vertex or face. This allows very complex comparisons, and operations in the third dimension can create dimensions consistent with already existing elements of the sketch.

The responsibilities for manipulating points and edges on the one hand, and faces on the other hand is distributed between the move and rotate tools, and the push/pull tool. Limiting the manipulation of the third dimension to perpendicular movements allows this complex functionality to be handled with surprising ease. Cooper and Reimann (2003, 314) noted in their discussion of interaction techniques that „because the application is focus on architectural sketching, not general purpose 3-D modeling … the designers were able to pull off a spare, powerful and simple interface that is both easy to learn and use".

Google integration created a further layer of tool architecture: apart from providing a web-accessible library with pre-defined and user-contributed objects, the combination with Google Earth positions SketchUp as an application that provides three-dimensional models as content for Google Earth that uses the model data to overlay renderings of architectural landmarks on satellite imagery.

Table 12: Minimal Aspects highlighted in SketchUp

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: Compared to other modeling packages, SketchUp is functionally reduced—mainly due to its focus on simple and quick model sketching. Consequently, rendering functionality is less sophisticated than in competitive products, e.g. 3D Studio or Maja.

Structural Minimalism: The number of tools in SketchUp is minimal, and only their combined use allows the user to initiate complex manipulations of models. SketchUp is not optimized for a specific use case, its comparably simple access structure is created by its architecturally minimal design.

Architectural Minimalism: The ease-of-use that defines SketchUp's initial impression—especially for novel users—is created by a clear distribution of responsibility between tools whose individual effects are easy to understand and predict. The almost seamless introduction of the third dimension that builds upon conventional 2-D drawing tools becomes possible through SketchUp's introduction of the push/pull tool.

Compositional Minimalism: On the one hand, SketchUp's use is clearly defined by its functional limitations, and by its special area of competence: it is often used by model builders to quickly draft ideas, or by architects, who are less interested in perfect modeling than in modeling speed and variability. On the other hand, it can be integrated in different ways in production processes leading to more refined models: its ability to im-

port pictures makes it a powerful tool for converting floor plans to three-dimensional models, and its ability to export models can place it at the beginning of a production line. The integration with Google Earth as a viewing tool adds a different layer of use as SketchUp can create content that is integrated into Google Earth's satellite and aerial imagery rendering, thus much improving the realism factor.

5.3.4 Apple iPod

After the initial success of the Apple II, and later the Apple Macintosh, Apple computers had to weather a number of crisis, as computers were priced too high for the average consumer (comp. Carlton, & Kawasaki, 1997, 103), and market shares were steadily declining (Burrows, 1997). CEO Steve Jobs had left, and later rejoined Apple, having founded NeXT and Pixar in between, and design again became an important aspect of Apple computers—most visible in the colored iMacs that heralded a new design fashion (Deutschman, 2001). When he introduced the iPod in 2001 (Jobs, 2001), the company's change from a software and hardware producer into a purveyor of fine media began. Today, Apple sales and revenue depend on the iPod series more than on its computers (BBC, 2004; Martell, 2006).

Description

While the initial iPod was already comparably simple, the introduction of the scroll wheel as an interaction device made navigation through its menu system much more pleasant: users could rotate first a plastic disc, then only their fingers on the iPod surface, and were thus able to rapidly browse their music library (Buskirk, 2004). With the clickable scroll introduced in 4th generation iPods and the iPod mini, separate buttons for playing and skipping could be removed, further simplifying the interface (Figure 74).

Figure 74: The Apple iPod (3rd and 5th generation).

The basis of the iPod's simplicity is the organization of the music (and later photo) library that is done *before* the music is downloaded to the music player. Without this

organization by artist, album, and, most importantly, by playlist[35], the wheel interface would still have a difficult time making scrolling through thousands of songs a pleasant experience. Instead of letting users organize their music on the iPod, Apple introduced the iTunes software for these tasks (Figure 77, Siracusa, 2001; Apple, 2001; Levy, 2006a).

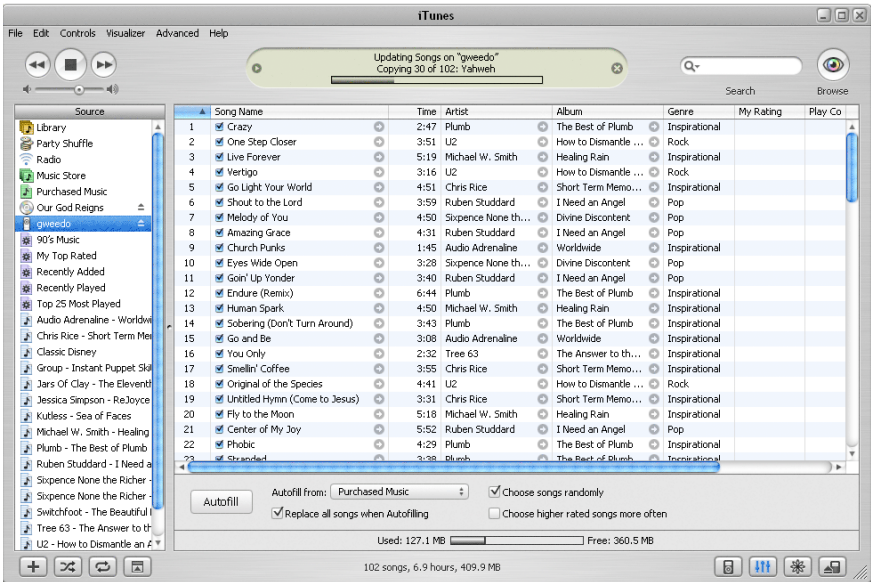


Figure 75: Apple iTunes is used to generate playlists, and fill the iPod with music.

iTunes can import music from CD, it automatically tries to retrieve the artists' names, album and song titles, genre, and even cover artwork from an Internet database. It can also be used to buy music from Apple's online music store. Individual songs can be rated by the user. Playlists can be generated by hand, or using automatically updated filters such as „was played recently", or „was rated highly". This makes iTunes a very versatile application that can provide quick access to even very large music libraries (Norman, 2005), and has made iTunes the basis of new social practices for music sharing (Voida, Grinter, Ducheneaut, Edwards, & Newman, 2005).

However, Apple did not stop there—parallel to introducing more sophisticated iPods that could also play small video clips (Levy, 2006b), in January 2005 they introduced the iPod Shuffle (Apple, 2005). The iPod Shuffle was stripped of most the features of its solid state MP3 competitors: it has no display, and no means of navigating through playlists. It can only sequentially skip through an existing playlist using a „next" and a „previous" button, or it can randomly skip through the playlist (ibid.). This reduction intended to shift the task of composing a playlist from a device that is not good at doing it—the flash memory player—to a device that has far better physical conditions—the computer running iTunes (Jobs, 2005).

35. Users can customize the main menu of their iPod to adapt the menu structure to their preferred organization system (Kiljander, 2004, 153).

Figure 76: The 2006 2nd Generation Apple iPod Shuffle.

This division of labor between a device that is only used for playing, and a device that is used to compose the playlist for the next hour(s) of listening, is reducing the requirements for the iPod Shuffle to an extreme minimum. Apple's designers were thus able to design a MP3 player that is „smaller and lighter than a pack of gum" (Apple, 2005). Although Apple openly communicated that the iPod Shuffle actually had less features than its competitors, the product was an immediate success, sparking reports of „mass hysteria" (Terdiman, 2005), and capturing a 58% share among flash music players in less than six months (Gibson, 2005). The second generation of this product built upon the qualities of its predecessor. It added no features to the player itself—which became only slimmer and more elegant in 2006. Instead, software features were added to iTunes that e.g. helped to increase the variation of songs synchronized with the iPod shuffle.

Discussion

The minimal perspective leads directly to the evolution towards the iPod Shuffle. However, other trends accompany this reduction and specification of the playing device. The iPod video demonstrates an important lesson for integration: choose the right time and the right modality, and do only what you can do well.

Apple's range of iPod models, particularly the iPod Shuffle, demonstrate the effectiveness of distributing a fairly complex task across two devices. This suggests that designs fitting Norman's notion of appliances (Norman, 1999a), denoting *functionally minimal* devices devoted to a single purpose, are possible if they are integrated within a context that shares functionality with the appliance. *Architectural minimalism* is thus a necessary prerequisite for *functional minimalism*, and only the focus on the ‚intelligent' context, or the ‚single-purpose' device determines the relevant notion.

Table 13: Minimal Aspects highlighted in the Apple iPod series

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: With the exception of the iPod Shuffle, Apple's iPod series did not publicly aspire to reduce functionality—although e.g. video features were only introduced after a satisfying user experience could be assured, both by technical means and regarding the availability of appropriate media. The iPods selling point was thus

never that it could do more than competitive players, but that it would deliver a better use experience. Apple positioned the iPod successfully to provide a ‚natural' way of listening to music etc., while special needs, e.g. recording functions, were left to third-party suppliers—even when the technical possibility was already implemented in the manufactured device.

Structural Minimalism: The patented clickwheel technology that evolved from the turning wheel of the first-generation iPod provides fairly adequate means of accessing items in long ordered lists, which are frequently accessed in the large music libraries that are commonly stored on iPods. Still, the menu structures in the iPod are deep, and not easily navigated.

Architectural Minimalism: All iPod players were designed keeping in mind that their screen space would be relatively limited, and that a personal computer would still offer better means to structure a personal music library. The iTunes software thus takes an important position for the iPod players as it functions as a second tool that creates the use experience together with the portable player. The iPod Shuffle is the extreme example for this strategy—a device strictly devoted to playing or shuffling pre-compiled playlists. Apple also developed a business model based on this differentiation, using iTunes not only to load new Podcasts, but also to access its music store, which in turn became the most successful venture in digital music so far.

Compositional Minimalism: Although the iPod is first and foremost a portable music player, some degree of *compositional minimalism* was important for it to become such a huge success. Factors that contribute to the many different uses that people find for iPods apart from carrying and listening are its many different models that cater for specific needs, from the music collector to the casual listener, and the modular architecture that sparked a „massive number of iPod accessories" (Enderle, 2005). Compatibility with this huge market for accessories is a further hurdle to overcome for competitors.

5.3.5 Web 2.0

According to general agreement, the World Wide Web was born in 1989 when Tim Berners-Lee and the often forgotten Robert Cailliau worked to create an information management system for in-house use within CERN, the world's largest particle physics laboratory. Since then, it has grown, changed, and developed (Berners-Lee, & Fischetti, 1999; Gillies, & Cailliau, 2000), and today has become one of the most important tools for information workers. Most work activity is interwoven with browsing on the Web (Brown & Sellen, 2001).

While the amount of information on the Web has grown enormously, the original concept of creating new value by linking distributed information (Berners-Lee, & Fischetti, 1999; Cailliau & Ashman, 1999) has become less important. The Web became centralized as Web sites offered specific information, or services, and users were happy to consume (Koiso-Kanttila, 2003); this went along with a development of Web clients to *browsers* that are used only to access, not to share information (Quint, & Vatton, 2005). However,

although users were unable to create links, or place pages on Web sites, many sites began to take in user input to increase the attractiveness of their services. Often, this followed the pattern set by Amazon.com with its user comments: the primary information was provided by those operating the server, and secondary information, e.g. reviews or recommendations, were added by unpaid users.

The term *Web 2.0* builds upon the original idea of a collaborative space, and extends beyond it as it is used to denote business models as well as systems for ordering information. Although the version number suggests a uniform effort for consolidation, or even a standardization, there exists no such movement (Shaw, 2005). The definition Tim O'Reilly[36] gives in *What Is Web 2.0* (2005) is consequently based on a variety of example applications (Table 14).

Table 14: Defining the Web 2.0 phenomen (O'Reilly, 2005)

Web 1.0	**Web 2.0**
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
mp3.com	Napster
Britannica Online	Wikipedia
personal websites	blogging
evite	upcoming.org and EVDB
domain name speculation	search engine optimization
page views	cost per click
screen scraping	web services
publishing	participation
content management systems	wikis
directories (taxonomy)	tagging ("folksonomy")
stickiness	syndication

While O'Reilly continues with a number of terms that are connected to the phenomenon he tries to describe (Figure 77), many of these terms seem to address specific applications, or at best individual developments in the Web, rather than creating a unified definition for a Web 2.0—while some aspects such as „the perpetual beta" or „rich user experience" will apply to most other software, as well (comp. Orlowski, 2005). The term Web 2.0 is defined as an umbrella term for all new developments on the Web—a wide definition of the term even includes a range of dynamic behavior of Web pages, namely the use of AJAX technologies[37].

36. Being somewhat of a Web 2.0 celebrity, O'Reilly was involved in coining the term (which has recently been reserved as a servicemark for CMP, a company holding the Web 2.0 conference series together with O'Reilly publishers).

37. This is yet another fashionable term describing the combination of manipulating a documents document object model (DOM) using JavaScript with the XMLHttpRequest method that is capable of streaming data. This allows Web pages to be updated after they are loaded, requesting additional

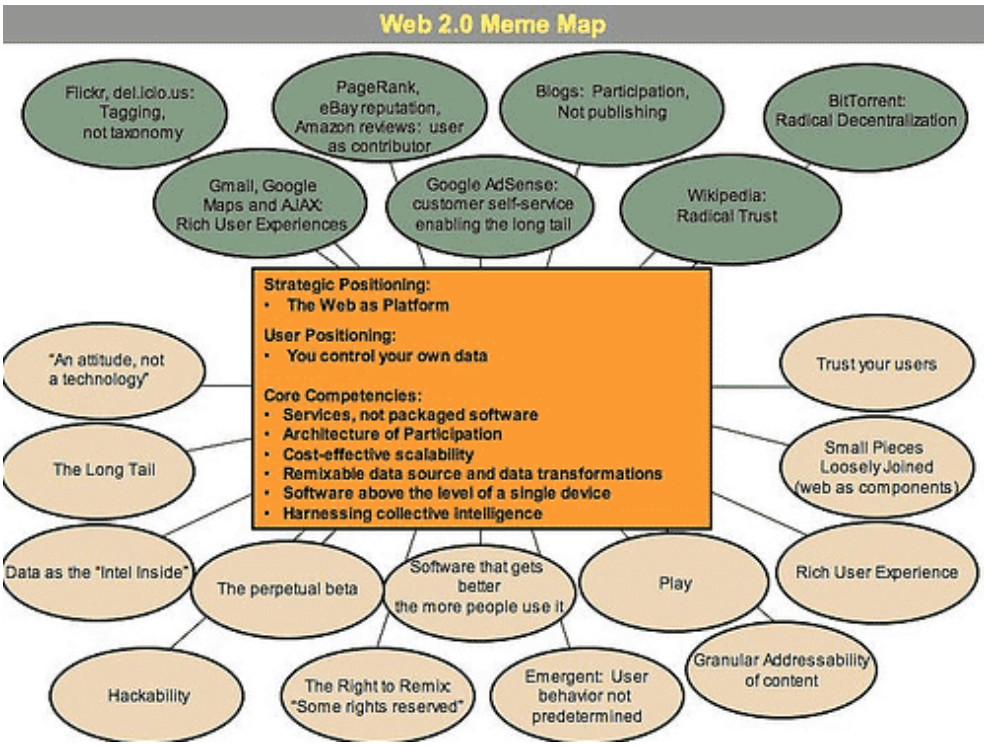


Figure 77: Tim O'Reilly approaching a definition of qualities of the Web 2.0.

After even this superficial examination, evidence suggests that the term Web 2.0 is not well defined. Its translation as „The New New Age" (Carr, 2005) suggests that the term is used to capture all the hopes that were already connected with the original Word Wide Web. For companies, it is a convenient marketing term, suggesting modernity and technical proficieny.

Analysis

Nonetheless, the term Web 2.0 is used for this discussion, although the scope will be limited to a single development within the World Wide Web: the novelty shared by most of the companies profiting from the Web 2.0 hype is the unselfish provision of data to other services, and the intake of data from other sites. The „First Web" used links to connect information archives, the „Second Web" uses links to connect applications. Consequently, the following analysis will focus on the aspect of increasing interaction of individual applications.

Compared to traditional desktop applications, the typical Web 2.0 application is functionally very reduced. An often cited example is Basecamp, a project-management tool provided by 37 Signals, which is built around a shared to-do list. Basecamp is simple[38], agile

information from the server side on demand—creating a much more dynamic behavior than possible before. (Garrett, 2005)

38. Jason Fried describes similarities of 37signals's philosophy to *functional minimalism*: „All of our products are about doing a few simple things well and leaving out the rest. We like to solve the easy fifty to eighty percent of people's problems and forget about the rest. You'll never make those people happy, you'll never

(it was built in four months with 2.5 programmers, Fried, 2004), and a visually attractive interactive Web application. However, Basecamp is also closed, allowing little to no data export—and although it features less functionality than e.g. Microsoft Project, all functionality for managing the project comes from the Basecamp site.

By contrast, other popular Web 2.0 sites, such as 43things.com del.icio.us, digg.com, flickr.com and Google Maps deliver only partial functionality. These applications do not only use the Web as a technical platform, their usefullness increases with the integration of other services: 43things allows you to input plans for your future, which becomes interesting as it connects you to other people, and their websites. Del.icio.us is an online bookmarking service that files your bookmarks under free-form tags that also allow you to access the bookmarks of other users. Digg.com is a technology news website that combines social bookmarking with non-hierarchical editorial control; additional value is delivered as users import stories from Digg.com into their own Blog with the keypress of a button. Flickr.com is an online digital photo sharing service that also allows you to create picture sets, tag pictures and image areas, and access other people's photographies; it became extremely popular as it is very easy to use Flickr to host your pictures and integrate them in your own Web site or Blog. Google Maps is a service that serves maps and satellite images to other sites, allowing the generation of localized services; simple examples are „memory maps" that import Google imagery into Flickr to collect childhood memories of neighborhoods[39], and overlaying a rising sea level with satellite imagery to simulate global warming effects[40].

The value delivered by these applications comes through their interaction with other Web services. Many small applications that interface to each other are thus able to deliver functionality that is not provided by a single service. This is so easily done that many an individual's Web page spring into life delivering yet another combination of different services—a phenomenon that has become popular under the name *mashup*. Vint Cerf explains: „There are creative people all around the world, hundreds of millions of them, and they are going to think of things to do with our basic platform that we didn't think of. So the mashup stuff is a wonderful way of allowing people to find new ways of applying the basic infrastructures we're propagating. This will turn out to be a major source of ideas for applying Google-based technology to a variety of applications." (Perez, 2005)

Discussion

The notion of *Web 2.0* is still very young, and not clearly defined. If it is understood as the functionality created by the interaction of Web applications, it presents a case for *architectural minimalism*—the multitude of services, and enthusiastic users indicate that a combination of individual functions promises great power. Technically, the Web 2.0 is dependent on a multitude of open standards, and financially, the majority of Web 2.0 appli-

---

totally nail that solution. Everyone always needs something a bit different and we don't go after that." (ibid.)

39. The memory maps group collects examples at http://www.flickr.com/groups/memorymaps/.

40. http://flood.firetree.net/

cation depends on individuals or companies providing their Web services without a monetary compensation. The question of who is paid how much for what functionality is not yet decided. An example is the multitude of services that build upon Google Maps—while Google currently provides their service free of charge, they are evaluating whether they will eventually switch to an ad-supported scheme, charge money for their services, or can make enough money on the data gathered from the requests (Sixtus, 2006).

This marks one of the problems of the Web 2.0 that are also applicable to architecturally minimal systems in general: as functionality is closely intertwined, data has to be exchanged between individual tools. In the Web, this currently works by giving one service the password for an account on another service; this requires very trusting users.

Table 15: Minimal Aspects highlighted in the notion of Web 2.0

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: A typical, but not obligatory attribute of „Web 2.0" applications is their focus on a limited set of features (e.g. bookmarking, sharing photographs). While this focus is partly grounded in the early status of many applications, functionality is often distributed between different applications, making a specialization possible.

Structural Minimalism: The Web 2.0 is marked by a fragmented interface—it is a challenge in its own right to keep track of the different evolving applications and find out which best suits the task at hand. Once a user has decided which service she needs, that service's access structure can benefit from the *tool-character* of Web applications.

Architectural Minimalism: An integral part of the Web 2.0 definition is its integrated nature. Applications draw on the functionality, and on the data, of other applications to together provide sophisticated interfaces to the user. This allow the individual applications to concentrate on a single functionality (e.g. delivering pictures for flickr, or maps for Google Maps), and places the burden of integration to the author of mash-ups. Through individual combination of services, users can quickly create new applications by combining existing services in form of mashups. A peculiarity of these Web 2.0 interfaces is, however, that although the application can be quickly generated and reconfigured, and users can choose between a large number of possible combinations to find the one catering to their individual tastes, the original services are often no longer identifiable to the end user.

Compositional Minimalism: The Web 2.0 does no longer only provide ready-to-use applications that one finds, personalizes, and uses. Instead, the metaphor of the application is at least partially replaced by that of the service, and users are gradually beginning to create their own applications using the functionality provided by others. The service provider thus only concentrates on providing a very specific functionality, and does not determine the context of its use. It is still questionable whether a transfer of

this concept to non computer-savvy end users can be managed, or whether the phenomenon is to stay with the geeks.

5.3.6 Word Processing

Introduction

A small german company, Brüning & Evert Softwarepartner GmbH, produced in 1986 one of the first word processing applications for the german market, running on Apple Macintosh computers. Already in 1989, this software integrated office and publishing tools: Ragtime 3, as it was now called, quickly captured a share of also the french market. In 1999, Ragtime 5 was released for Microsoft Windows, and in 2003, a version for Mac OS X became available (Ragtime, 2005).

Analysis

Ragtime never aimed to compete directly with Microsoft Office, unlike other products such as StarOffice/OpenOffice.org, KOffice, or Abiword. What makes it an interesting example here is its very different philosophy: it does not try to accumulate features from domains, such as spreadsheets, drawing applications, or DTP layout software, and integrate these into a simple word-processing software. Instead, the basis for integration is not the continuous text, but the layout of the page: In Ragtime, frame elements are positioned on the page. A frame's type defines both its rendering and applicable tools—there are frames for word processing, frames for spreadsheets, for diagrams, for bitmap and vector graphics, and for movies, pictures and sounds. When a frame is selected, the accessible functionality is adapted to the content of the frame: click and selection semantics change, context menus contain only useful commands (some general, and some content-specific), and toolbars are exchanged to provide useful functionality.

Figure 78: Ragtime 3 introduced the different content types still in use today.

Ragtime exemplifies the combination of different tools within a single application. Each single tool is much simpler than its competitors, e.g. the spreadsheet tool cannot create diagrams by itself like e.g. Excel, and the word processor cannot create tables. This is compensated by a close integration of the tools. Ragtime's basis is the spatial layout of content areas on the virtual page. Each area is assigned a content type (Figure 79) that determines the tool responsible for rendering and manipulation.

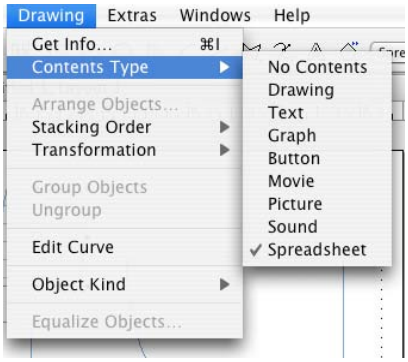


Figure 79: Content type selection in Ragtime 5

The data in individual tools can be linked to create sophisticated renderings: the graphing tool takes numerical input from a spreadsheet, automatically updating the graph when the numbers are changed. The layout functionality uses direct manipulation for the content areas, much as special DTP programs do. This makes it possible to graphically combine e.g. a spreadsheet table with text. Each tool is responsible for strictly delimited tasks, and can thus be functionally very simple.

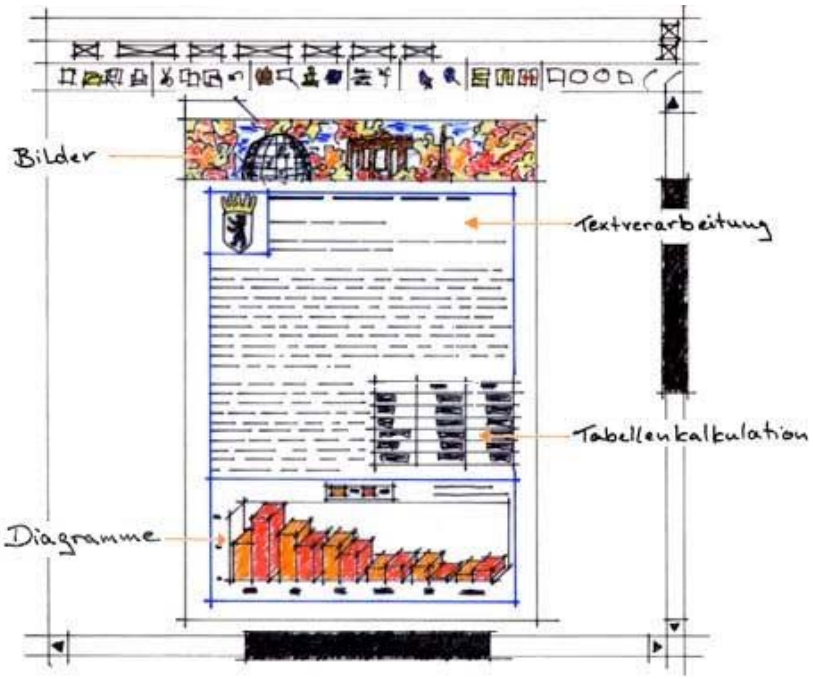


Figure 80: Placement of different content types on a page. Ragtime 5 documentation.

In its latest version, Ragtime 6, some of the unique design decisions are watered[41]: the word processing component of Ragtime was redesigned for greater compatibility with the market leader, Microsoft word. Consequently, it is now possible to integrate pictures and table directly within the word processing component; although this duplicates functionality within the product, and overrides the distribution of tasks between tools, the underlying motivation is clear: a greater external consistency with other word processing applications (Ragtime, 2006b).

Conclusion

Ragtime demonstrates that word processing is possible without competing with the de-facto standard Microsoft Word for feature-richness. Instead, the functionality that is highly redundant within Microsoft's office suite—with Excel and Word both being able to create tables, graphics, and stretches of formatted text—is distributed over a number of tools. These tools have a fixed, and very limited field of responsability, and are consequently much simpler than their Microsoft counterparts.

This division of responsability is communicated throughout the user interface: the user has first to choose the type of content (defaulting to text) before she can start manipulating it. For this, she is rewarded with a concise interface providing a set of functionality that is tailored to the content type. The contextual tabs of Microsoft Word 2007 is a step in the same direction, yet in Ragtime, the entire interface is based upon the provision of different tools for different tasks. As the tools are chosen based on the type of area focused, the change between tools does not create a cognitive burden for the user.

Ragtime 7 dissociates itself somewhat from this clear-cut approach; a possible reason is that to persist in a commercial environment, compatibility with the Word file format is a vital issue. As Word is able to mix tables, graphs and text in the layout flow, Ragtime might have been forced to comply.

5.3.7 Refining the Notion of Architectural Minimalism

Architectural minimalism can refer to the interaction level, SketchUp being an example; it is here closely connected to structural minimalism. It can also refer to the interaction of specific, functionally minimal tools. In both cases, the whole is simpler than the sum of its parts. Lego demonstrates a further, important aspect—complexity is created by the combination of simple building blocks; this example, however, lacks a good demonstration of the interaction of different tools as most Lego pieces are very similar.

While Sketchup and Ragtime seem to lie worlds apart, they are at two ends of a line that is completely covered by the term architectural minimalism: In Sketchup, atomic actions are applicable to parts of the drawing. Combined, they allow complex manipulations. In Ragtime, the tools are much more powerful, yet the combination of the functionality of

41. Ragtime also canceled the unique distribution model that allowed for free copies for private use. (Ragtime, 2006a)

individual tools through placement on a page is still what delivers value. In both cases, the functionality is deconstructed in a manner that lets the user perceive less complexity than hiding in the individual tools taken together: the whole seems simpler than the sum of its parts. Somewhere in the middle of that line lies object-oriented drawing software, such as Adobe Illustrator or OmniGraffle[42], where individual objects can be manipulated by different tools; some of these are dependent on the type of object (e.g. font selection), others apply to all types of drawing objects (e.g. manipulation of size or orientation). Full object oriented environments, such as the Smalltalk-based Squeak (Ingalls, Kaehler, Maloney, Wallace, & Kay, 1997), also draw part of their attractiveness from the fact that a multitude of simple tools exist that can be applied to objects, and combined to deliver complex functionality.

The tools in an architecturally minimal design are visible to the user. If their use is linked to the type of object being manipulated, the functionality provided by the interface is changed as an object is selected. Usually, modes have been used to allow the user to change functionality: in a drawing program, the user could either move objects, or change their color. Modal interfaces have been heavily criticized as they can cause handling errors when the user does not remember which mode he is in (Raskin, 2000b). Coupling the functionality with the selection of objects can for some applications replace modal behavior, as the outcome of an action is not determined by a system state that is difficult to discern, but by the focus of activity—of which the user generally is aware of.

Architectural Minimalism, and the resulting focus on creating visible tools is reflected in Züllighoven's „tools and materials" approach (Züllighoven, Bäumer, & Bleek, 1998; Züllighoven, 2004) to object-oriented software construction that has been implemented in form of the Java framework JWAM (Breitling, Lilienthal, Lippert, & Züllighoven, 2000). Underlying this approach is a desire to create useful and useable software by relying on the existing distribution of work in an office environment: „the software product should represent the main concepts of the application area" (Lilienthal & Züllighoven, 1997, 36). Lilienthal et al. aim to create a „close relationship between the tasks and concepts of the application domain and the components of a software system. This relationship allows the users to recognize their specific means and objects of work and enables them to fulfill their tasks by *individually organizing work as the situation demands it.*" (ibid., 36, italics mine) This demonstrates that Lilienthal et al. aim to support the user's interpretation of a task by providing separate tools at her disposal. The „tools and materials" approach tries to identify materials to separate functionality accessible to the user from „»pure« application domain functionality" (ibid., 37). Tools then try to bundle functionality in meaningful—and universal—units: a tool's quality depends among other factors on the number of contexts in which it can be applied (Züllighovern, personal communication).

42. Information about Illustrator is provided by Adobe at http://www.adobe.com/products/illustrator/, OmniGraffle is described at http://www.omnigroup.com/applications/omnigraffle/.

In a minimal architecture, the overall functionality is decomposed and distributed between tools. When the user is allowed to change the visibility of these tools, the user interface of an application can be *configured* by users to better meet their task requirements—Eclipse.org is an example for a popular IDE that also allows users to add third-party tools in form of plug-ins, an increasingly popular concept. When these tools are exposed not only to direct manipulation, but also offer a scripting interface, the added value created by combining the individual tools can be automated, and new tools be created. Although this *end-user programming* is not as widely spread, Automator is an interesting approach that build upon Apple's functionally simple applications and their common APIs to create a graphical editor for use patterns.

A continuous spectrum is spanned between the loose coupling of applications via import and export functionality, drag & drop, and copy & paste mechanism, integration of services in another application, and the deep interconnectedness created by linking existing applications. The combination of different tools can also work on the level of interaction, as demonstrated by the SketchUp application—and the integrated use of different tools can pursued even further as the selection of tool and application to a graphical object are parallelized using two hands, e.g. with the tool glass concept, whose practicability has been demonstrated in a free graphical Petri net editor (Beaudouin-Lafon, 2000).

The application of architectural minimalism has its limits: Whatever the reason was for Ragtime to integrate duplicate functionality in their text component, it is obvious that a division of functionality becomes more difficult as the individual tools are developed and acquire additional features; eventually, overlap of functionality will occur. UNIX operating systems can serve as an example for some pitfalls for architecturally minimal systems: Traditionally, UNIX systems have employed a multitude of different tools with a very focused functionality to solve complex tasks by composition: the complexity is reduced by identifying subtasks that can be handled by simple tools, such as grep, sed or awk, and then these tools are combined by pipe-lining the output from one tool to another for further processing. Thus, repeated actions that would require many commands can be automized; this *shell scripting* has become a synonym for harnessing the power of the command line (Robbins, 1999).

However, the number of UNIX tools is huge, and to apply the right tool to the task, one has to learn „why each one is there, how to use them by themselves, and in combination with the other [tools]" (Robbins, 2005), a typical task for expert users. Furthermore, the tools themselves have changed to incorporate functionality previously reserved for other tools, and introduced inconsistencies between different flavors of UNIX. As Robbins and Beebe (2005) put it in *Classic Shell Scripting*, „with shell scripts, you can combine the fundamental Unix text and file processing commands to crunch data and automate repetitive tasks. But beneath this simple promise lies a treacherous ocean of variations in Unix commands and standards". This functional overlap is potentially a critical problem for architectural minimalism—if taken to an extreme, each individual tool becomes very complex in itself, thus making the system of tools that a user needs even more complex than a single all-purpose tool would need to be.

5.4 Compositional Minimalism

The three previous notions of minimalism focus on doing more to create less, on the designer's work of selecting, structuring and partitioning functionality in a design. Compositional minimalism poses the question where a designer can do less to let users do more—how tools must be created that will be usable in many different situations. This paradoxical requirement is illustrated using two examples from everyday life: the appropriation of architecture and the ubiquitousness of the Post-It note illustrate the different aspects of compositional minimalism, namely that products will always be changed by their use, and that concepts can be so successful that they appear in contexts where they were never expected.

5.4.1 Old Buildings Learn

Architecture is a favorite place for software engineers, or software „architects", to visit in search of helpful analogies; the most famous analogy resulted in various pattern movements, first in software engineering, and later in usability engineering. Another school of thought has not (yet) found its way from the realm of architecture to the design of interactive systems: Humans have for a long time created their own dwellings, and only comparably recently, the profession of architects has formed to plan and engineer buildings (Kostof, 1977). But even now, home owners are not content to simply live in their houses, they are prone to change plans. Architecture has fittingly been described as „the relation between an object and its occupant" (Hill, 1998, i). As Steward Brand (1995) put it, buildings *learn*; after they are built, they are adapted and then adopted—through modifications, minor or major, they become what their ‚users' want them to be. A 1935 photograph of Johnstown, Pennsylvania (Figure 81) illustrates that „backyards are where the action is: … nearly every house shows signs of additions and changes in the back" (ibid., 197). Adaption, and visible change is the rule, not the exception.

Figure 81: „Backyards are where the action is": residents usually modify and adapt their houses.

According to Brand (ibid.), buildings can be separated into different layers. Although these layers are interdependent—a change in services might make necessary changes in structure, inner layers can often be modified decisively, with only minor modification to the outer layers. He notes, however, that specifically the *services* layer of houses is very expensive to change afterwards and, if planned in an inflexible way, can interfere with useful modifications.

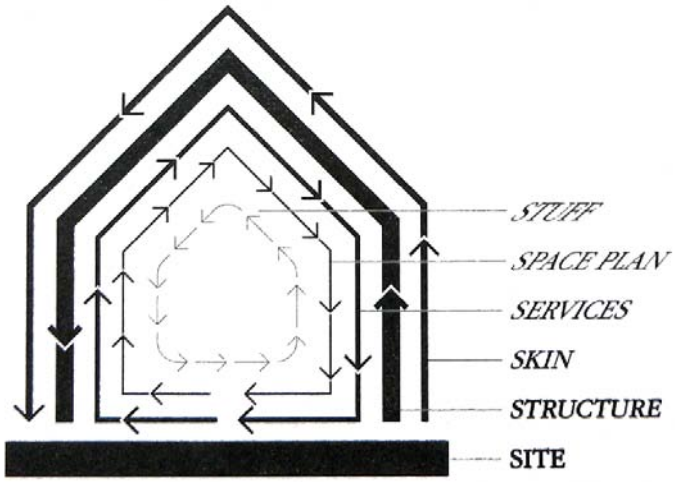


Figure 82: The structure of buildings can be examined—and changed—at different levels.

As an example for a well-planned house that allows many different uses, he does not pick a stylish or modern building—his example is the standard American municipal building from the 1960s. Through standardized design, equal spacing, and free-standing walls on the inside, the space plan could be changed almost at will, and houses originally planned as municipal offices now often serve as a library or kindergarten (see Figure 83).

Figure 83: The ‚universal' public American building of the 50s. (Brand, 1995, 154)

As the architect refrained himself from making too many assumptions about the use of the building—the chimneys are at the side of the building, as are all supporting walls—and as the focus of possible uses was very broad from the beginning, the unassuming ar-

chitecture of municipal buildings has helped to create a space with many uses for its inhabitants and visitors.

Moreover, adaption is so much a part of dwelling that it has become an indicator of coziness, and thus turned into a streak of fashion—and there are houses being built that look as if they had been added to by generations (Figure 84). However, houses that are made complicated by artificial ‚additions' do not only fake adaption, but also hinder real adaption to take place: as the structure is unnecessarily made complicated by bays and dormers that serve no obvious function, it becomes more difficult to serve emerging needs by adding further spaces to the house (ibid., 201).

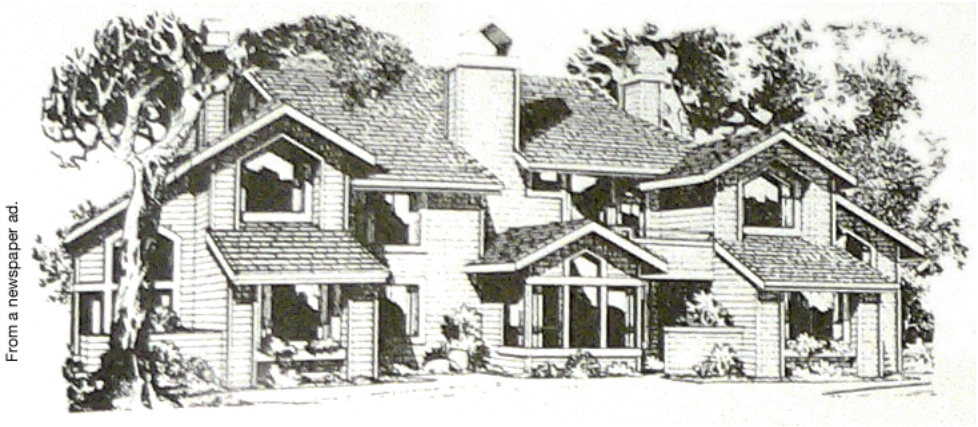


Figure 84: An artificially complicated building with a vintage look. (Brand, 1995, 201)

5.4.2 A Sticky Story: the Post-it Note

A short digression might clarify the nature of the two following examples, Email and Microsoft Powerpoint: tools developed for a very specific purpose can gain wide acceptance and their use can spread exponentially into ubiquitousness. Perhaps the best example for this is related to interactive systems only insofar as it sticks on most computer screens: the Post-It note[43].

Post-it notes may be considered as literally „god-sent". In the early 1970s, Art Fry was in search of bookmarks for his church hymnal that would neither damage it nor fall out. Fry worked at 3M. He noticed that a colleague, Dr. Spencer Silver, had developed an adhesive that left no residue after removal and could be repositioned. Fry took some of Silver's adhesive and applied it along the edge of a piece of paper. This solved his church hymnal problem. Fry soon realized that his „bookmarks" had other potential uses. As he began to leave notes on a work files, coworkers started dropping by, seeking „bookmarks" for their offices. These „bookmarks" were a new way to communicate and to organize. 3M Corporation crafted the name Post-it note for Fry's bookmarks and began production in the late 70s for commercial use (BBC, 2000).

43. The name „Post-It Note" is a trademark of 3M Inc. patent.

In 1977, test-markets failed to show consumer interest. However in 1979, 3M implemented a massive consumer sampling strategy, and the Post-it note took off. Today, Post-it notes and their copies are scattered across files, computers, desks, and doors in offices and homes. The church hymnal bookmark has changed the way most information workers everywhere on this planet work every day.

Discussion

Buildings are changed as their owners change; they also change as their function shifts, e.g. when kids arrive. Allowing for this change is a necessary quality in architecture that tries to support humans in their lives. Adaptation of buildings is supported by a layered architecture that differentiates those parts that cannot be changed easily (e.g. load-bearing walls) from those that are modified more easily. The design of the more static layers then tries to maximize the possible arrangements of the more dynamic layers. For architects, it is vital to consider the context of a building, yet it is impossible to foresee the use of a building in dozens of years from now; the longevity of architecture renders ineffective approaches that try to define all requirements before a building is in use. For software, by contrast, future legacy issues are often not even considered in the initial design—while the software complexity exceeds that of buildings, and the accumulated experience for building materials in software engineering is much less.

The Post-It note makes a case for designs that target very specific requirements. Although it was designed for a single user, and a single task, it has gained world-wide acceptance. Almost unchanged, it can be used for a church hymnal, and for business communication. rare case. simplicity of its design—the removable adhesive being its only feature. That is marketed now even without the paper, effectively allowing anything light enough to be turned into a Post-It.

5.4.3 Email

One of *the* most fundamental revolutions that computing has caused is what Email has done to communication: instead of sending hand-written or -typed letters, or facsimile copies of the same letters to companies we trade with, or people we deal with, Email has taken on the function of communication for most of us in many situations (for an early analysis, compare Denning, 1982)—it has become ubiquitous (Berghel, 1997b; Ducheneaut & Bellotti, 2001; Tassabehji & Vakola, 2005), created new social spaces (Parks & Floyd, 1996), and grew so important that understanding email communication turned into a research topic for itself (e.g. Wilson, 2002, 126).

Description

Email use has changed over time, and as use scenarios broadened, the nature of communication has diversified. Initially, the cost of disk space prevented Email from being saved, and although some users treated it like written correspondence, closing messages with „sincerely", or „yours truly", users quickly developed an informal writing style; most early email mimicked oral conversation, with heavy use of exclamation points, repeated question marks, and capitalized words (comp. Lovejoy, & Grudin, 2003); John Seely Brown in

his plenary address at the first CHI conference in 1983 even told the story of a Xerox executive who was embarrassed about his spelling, and presented his solution: „We'll build a spelling de-corrector!" (Brown, 1983, cited after (Grudin 2003)). This informality caused suspicion in management, even for technology companies (Perin, 1991), and fear of productivity losses (Pickering, & King, 1992).

However, tools were invented that fixed spelling and grammar, and Email has long become widely accepted for more formal correspondence (Tassabehji & Vakola, 2005). While it is still used for private communication, it has become an irreplaceable work tool for many. Moreover, as email is so easily accessible, users started to employ email as a tool for tasks it was never designed for, e.g. task management and personal archiving (Mackay, 1988; Whittaker, & Sidner, 1996). And users are inventive—they use the sorted table emails are usually presented in to manage contacts and todo lists, send themselves reminders or found Web URIs (Jones, Bruce, & Dumais, 2001) that can be read from different computers, and use the archived emails as a big, searchable information storage. Although this demonstrates that few technical restrictions are placed upon users by email, and that for many, it seems to fulfill their tasks better than dedicated tools, it also creates the problem of *email overload* (Whittaker, & Sidner, 1996) that has led to many research efforts trying to devise clever replacements for traditional email tools that are tailored to deal with the added task inventory under the label of *personal information management* (PIM) (for a comprehensive overview, see Boardman, 2004, 40-48).

These efforts can be classified in those that aim to reintroduce specific tools with specific workflows to support specific tasks, and those that try to better support the appropriation of existing mechanisms by users. As the notion of compositional minimalism highlights, the latter approach builds upon the strengths demonstrated by Email as a tool. It is also likely that the efforts users already invest to tailor their tools to their task environment can be built upon; e.g. Bellotti and Smith (2000) report that Email takes not only a central role in personal information management, but also that users appropriate features of their Email tools, thus building embedded mechanisms to manage complexity.

Email is far from being an average application. It was one of the first computer applications making use of networks (Hardy, 1996), and is still the most important electronic medium to connect individual people. In its long history, it has been used for many different purposes—and for this been named not only „the killer application of the Internet" but also „a serial-killer application!" (Ducheneaut & Bellotti, 2001) Through the central role it already occupied in daily work, it attracted new tasks. As Whittaker et al. (2004) state, „Email's role as de facto task manager arises in large part out of its role as information conduit"—both incoming and outgoing information are often routed through email. Ducheneaut et al. (2001) suggest, „The network effect ensures that this tendency is infectious across a community or organization. Thus, personal information management is then embedded where it is most needed and accessible, that is, in the knowledge workers' new electronic habitat: e-mail." Email is used for managing tasks, often by reminders (Malone, 1983; Barreau & Nardi, 1995; Whittaker, & Sidner, 1996), and for archiving information (Kaye et al., 2006; Whittaker, & Sidner, 1996; Whittaker, Bellotti, & Gwizdka,

2006)—users know email as a stable, time-ordered and searchable form of information, and they also know they will use their email very frequently. Full-text search, sequential ordering and association with important contacts provide powerful access mechanisms that can be combined with manual categorization (Malone, 1983; Barreau & Nardi, 1995). Email is even used to keep contact information (Whittaker et al., 2004).

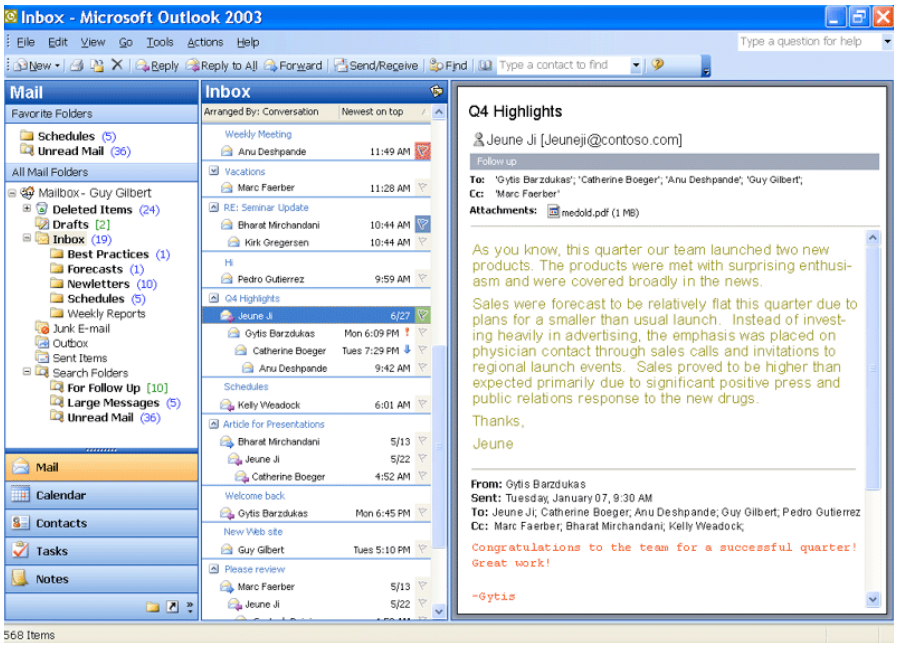


Figure 85: Email client with folders used for information management (MS product screenshot).

The broad field of tasks that email is used for is often seen critical. Problems include fragmentation, e.g. duplicate storage in email and file system that makes it difficult for users to locate information (Whittaker, Bellotti, & Gwizdka, 2006), and lack of dedicated support mechanisms for users' tasks, e.g. for contact management or time management (ibid.). However, from a minimalist standpoint, it is interesting to note that there is a discrepancy between research and use: On the one hand, many research prototypes try to provide sophisticated personal information management aids (e.g. Bellotti, Ducheneaut, Howard, & Smith, 2003; Bergman, Beyth-Marom, & Nachmias, 2003; Dumais et al., 2003). On the other hand, even in software that provides much of the proposed functionality, much of it is left unused: „The results from 33 Outlook users suggested that even our technologically savvy lab members often did not use the more sophisticated features of Outlook (e.g., only just over half use the Tasks feature and far less than half use Categories to organize their mail). Our less tech savvy lab members (a small minority) were especially unlikely to use these features or even to know about them" (Bellotti, Ducheneaut, Howard, Smith, & Neuwirth, 2002).

Discussion

Email seems to work well enough for most people; replacing email by dedicated PIM tools is made difficult by email's central position in the workflow. Boardman and Sasse

(2004) inferred from an empirical study that the everyday use of email discourages users from reflecting on their own behavior. They concede that many users will continue to use only basic functionality, and follow that „Future design work must take account of the variation in strategies by providing the flexibility to manage different types of information in distinct ways. For instance, tools should give users the ability to organize information as required, whilst not penalizing those users who do not want to organize." (ibid.)

A possible enhancement for email clients should thus adopt the form of optional tools; these tools must not create unnecessary, additional tasks for the user (Whittaker, Bellotti, & Gwizdka, 2006). Examples include both approaches that enrich the folder structure that exists in email clients through searching, sorting and tagging, and automatically categorizing email and providing alternative views (e.g. Bellotti, Ducheneaut, Howard, & Smith, 2003; Rohall et al., 2004; Danis et al., 2005; Whittaker, Bellotti, & Gwizdka, 2006), or the closer integration of external tools with the mail client. The latter approach is taken by Apple in MacOS 10.5 where the Address Book, iCal and the Notes application are being integrated into Mail 3 (Apple, 2006b).

Recent developments in electronic communication indicate that real-world use of a concept, such as Email, is also influenced by the public opinion that originates with its use. Email has lost some of the informality associated with it, because it has acquired a central role in the office workflow, and has thus turned into an archival medium. *Instant messaging* (IM) has begun to replace it in some contexts, and for some users (Lovejoy, & Grudin, 2003; Grudin, Tallarico, & Counts, 2005). This demonstrates that the adoption of software is not only difficult to plan for, but that it must also be considered a self-influencing dynamic system.

Table 16: Minimal Aspects highlighted in Email

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: Email has but a single function: that of delivering a message (basically an ASCII file with title, sender, and date information) to a recipient address over a network. It was later extended to transport binary files, first via uuencoding and later using MIME (Levinson, 1998).

Structural Minimalism: Due to the functional simplicity and the similarities with postal mail, Emails are easily understood as a concept. Email clients, however, are generally less simple as they add different filtering and structuring mechanisms to the base functionality. Structuring Emails in folders, sometimes with marking facilities, has become a very popular way of dealing with incoming and archived information, and generates a user-specific, yet often rather sophisticated ordering structure.

Architectural Minimalism: As Email is technically very simple, many other tools interface with Email as a outlet, either locally by starting the Email application with a set re-

cipient, title and text, or by sending out Email, e.g. from Web applications, to signify e.g. status changes. Email is scarcely used as an inlet, further adding to its central position within the daily workflow.

Compositional Minimalism: Although, or perhaps because Email is technically very simple, it has become the pivotal center of information management for most of todays information workers. As it is a permanently used, and almost always accessible tool, and as most information flows are anyways routed through Email, it took on responsibilities that were never planned to be implemented using mail: information management and storage, contact management, time scheduling, note-taking, project management, and a multitude of other important tasks are carried out using the simple means that email clients provide. This is *not* a bad thing. Instead, it shows how a versatile information tool can cleverly be adopted by competent users. Thus, it should not be first priority to replace Email as a tool for these tasks, but rather to intelligently interface to the existing practices as new tools are developed that try to solve the existing problems with Email.

5.4.4 Powerpoint

> It is a structure for following your intuition and your obsessions.
> It is the hyperfocused scribblings of the mad and the gifted.
> — David Byrne, Learning to love PowerPoint (Byrne, 2003)

Powerpoint, originally *Presenter* for the Apple Macintosh, was created in the early 1980s by Bob Gaskins (Parker, 2001). Since then, it has been bought and marketed by Microsoft, and risen to what is one of the most successful business software products to date. Experts who follow trends in presentation techniques estimate that Powerpoint is used to make an estimated 20 to 30 million presentations every day and has between 250 and 400 million users around the globe (Schwartz, 2003; Zielinski, 2003; Simons, 2004).

This vast number of users includes the manager who depends on Powerpoint slides to carry him through meetings, but also the private citizen who designs slideshows for her son's wedding. Pupils prepare powerpoint slides for school, and even church masses increasingly adopt the software to flash hymn lyrics, or accompany preaching (Parker, 2001). Public, the mechanics of scientific face-to-face exchange depend so heavily on Powerpoint that it seems not disproportional a conference was interrupted to thank Gaskins personally as the organizers learned that the creator of Powerpoint was in the audience (ibid.). Even the icons included with the software, androgynous silhouette stick figures, have become omnipresent as Powerpoint is also used for layout tasks (ibid.)—to create signs, web sites and printed material.

Description

Powerpoint is an application with a fairly limited functionality—at least, if one compares it with other office applications. By design, the application domain is fixed to slide shows, and thus all functionality is grouped around the slide view; with a simple outline editor, a structural perspective is provided in addition to the layout view of the slide. Powerpoint

can thus be considered an untypical mixture of functionality for text layout, object-oriented drawing and animation—it is possible to find many different uses for Powerpoint, yet in most areas, it has rather basic functionality.

It seems, however, that Powerpoint's functionality is enough for most users—it has become an immensely popular multi-purpose tool. Because it is so easily mastered, some users actually prefer to use Powerpoint to create not slides, but text documents, drawings, signs—and even art (Byrne, 2003): „PowerPoint is a hybrid: it does many things moderately well. Because it is a single program that allows you to combine simple layouts with diagrams and prose, it is both an ideal tool and ripe for misuse." (Brown, 2002).

Although it has a fairly complicated menu structure, introductions to Powerpoint suggest that users use the toolbars to easily access functionality (Haddad, 1999; Finkelstein, 2003). With a multitude of different tool palettes that partly appear on-demand, Powerpoint can be said to adapt itself to the users tasks, e.g. when the drawing palette, or the image palette appear as the users selects a graph or image object. Many common tasks are automatized, e.g. text is reformatted as content is added or deleted to optimally fit on the slide—this culminates in the implementation of an auto-content generator, which had originally been just an internal joke for management (Parker, 2001).

Powerpoint's functionality is often not sufficient for the task. However, it is still used as an outlet, with other applications delivering different types of content. For the task of creating slideshows, for example, Johnson and Nardi (1996) found that applications like Powerpoint are often not used in isolation. As its functionality is not sufficient to create e.g. different types of formulas for engineers, mathematicians, or chemists, or specialized drawings, many other tools are involved in creating a presentation.

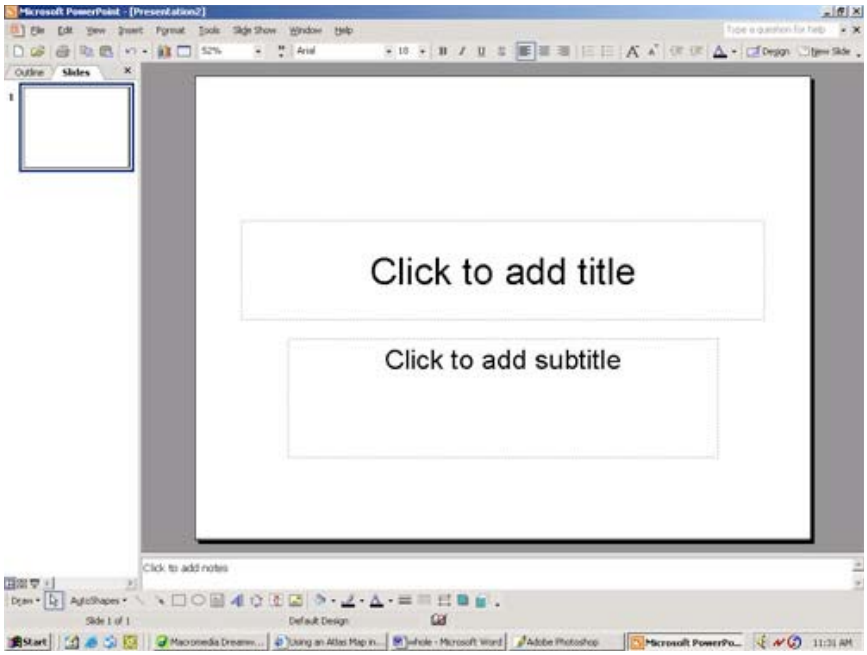


Figure 86: Microsoft Powerpoint.

Due to the multitude of presets that Powerpoint provides, it has a flat learning curve. It provides a skeleton for implementing the task of delivering a talk that is useful for many users as it enables them to do a fairly professional job without having special training: it helps structuring the material, suggests form and amount of presentation, maintains a clear layout, and ensures compatibility to most presentation environments (Mertens, & Leggewie, 2004).

Yet the provision of default layouts for slides that e.g. fit only 4-6 bullet points, and limit the title to about 3 words, has repeatedly been criticized. Critics find fault with „slide designs [that] have become more standardized, in large part because PowerPoint itself is used so pervasively". (Alley & Neeley, 2005) Powerpoint has even been made responsible for negatively influencing presentation style, and (scientific and business) communication as a whole (Brown, 2002; Tufte, 2006). Edward Tufte found a concise slogan for critique „Power corrupts. Powerpoint corrupts absolutely." (Tufte, 2003) Some research actually targets the question how the communication style that has been „damaged" by Powerpoint could be changed for the better. Alley and Neeley proposed the use of pictures and sentence-long headlines (2005) to avoid the need to create 3-word headlines for all slides. Although some doubt remains that their alternative approach will cater to a wider audience, and improve scientific exchange as a whole, it is certainly worthwhile to notice that the defaults in Powerpoint have had an immense influence on the design of both business and scientific communication—a composition that was originally conceived for a single context has influenced interpretations in different fields.

Discussion

The concept of composition implies that a design might be more or less strictly composed for a task. It is thus tempting to think that applications could be judged according to their specificity: a presentation application would be more specific than a word processor, and less specific than an application to draw chemical formulas for molecules. More specific applications would rate as less compositionally minimal, and vice versa. However, Johnson and Nardi found even in 1996 that „the task specificity/genericness of an application program … depends on several fairly independent software design issues. We (1) conclude that developing application software that supports all aspects of a task well is extremely difficult and (2) suggest an alternative approach that may be more fruitful: providing collections of interoperable tools and services." (Johnson & Nardi, 1996)

Johnson and Nardi come to the conclusion that no single application can deliver a functionality that is as diverse as that needed for the generation of presentation slides. Instead, they suggest that a „collections of interoperable tools and services" (ibid.) should be used to combine the necessary functionality. This highlights that in the case of an ubiquitous use of an application, it is difficult to meet the growing and diversifying demand for functionality with a monolithic architecture, and thus illustrates the link between architectural and compositional minimalism.

Table 17: Minimal Aspects highlighted in Microsoft Powerpoint

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: Powerpoint is a hybrid application, it offers minimal functionality in many different fields: it can be used to manipulate text, diagrams, and graphics—and it provides limited support for (interactive) animation. In each of these fields, more sophisticated applications exist, but Powerpoint is unique due to its combination.

Structural Minimalism: Powerpoint is both structurally very complex, e.g. in its rather unorganized menu structure, and very reduced as tool palettes offer rapid access to the rather limited functionality responsible for e.g. text formatting.

Architectural Minimalism: Powerpoint combines different tools, and each tool has only a limited functionality. In contrast to object-oriented drawing applications, and more specifically to SketchUp, there is little functional overlap in Powerpoint, i.e. few functions are applicable to different types of content. The combination is thus not on the level of interaction, but lies within the creation of a collage to create the visual experience of a presentation slide.

Compositional Minimalism: Due to its basic and general functionality, Powerpoint is used for a multitude of tasks very different from presentation. When used for presentation, however, its functionality is often not sufficient for generating very specific expert content. In these cases, it functions as a vehicle for presentation of context generated in other tools and is never used in isolation.

5.4.5 WikiWikiWebs

In the original proposal for the World Wide Web (Berners-Lee, 1989), the Web was envisioned as a place to share information, accessible to anyone. This was not going to happen—as Berners-Lee reflects a decade later, „although browsers were starting to spread, no one working on them tried to include writing and editing functions. … Without a hypertext editor, people would not have the tools to really use the Web as an intimate collaborative medium. Browsers would let them find and share information, but they could not work together intuitively" (Berners-Lee, & Fischetti, 1999, 57).

Although the Web that emerged in the 90s turned out to be site-centric, consisting of few major players who contributed information, and assigned a passive role as consumer of information to the user, the vision of everyone contributing to a world-wide information web has repeatedly returned to the stage: Whether it was called *authoring hypertext* (Carr, DeRoure, Hall, & Hill, 1995), *collaborative writing* (Hughes, Shewmake, & Okelberry, 1998), *global editability* (Iorio, & Vitali, 2005), or *The Writable Web*, it is still a prominent force in hypertext research—and as a topic it also appears at the World Wide Web conference series (e.g. Miles-Board, Carr, Kampa, & Hall, 2003; Uren, Shum, Li, Domingue, & Motta, 2003). As Bouvin (1999) summarizes various efforts for enhancing the Web, „the purpose of such … tool[s] is [to] help users organise, associate, or structure

information found on the Web … by a single user or in collaboration with others". However, all approaches that required users to install additional tools on their computers were unable to attain a critical mass of users, or reach a state of satisfactory stability[44]. So, until today, the Web of most users is still pretty much ‚unenhanced'.

More successful was the approach of server-side applications: here, not only in research, but also in practice has authoring returned to the Web. One important genre of user-writable sites are Web logs, or Blogs (Blood, 2002). A Blog is usually published by a single author, with reader being able to refer to individual articles, thus creating a Web of comments. While this describes the current state, „the original weblogs were link-driven sites. Each was a mixture in unique proportions of links, commentary, and personal thoughts and essays. Weblogs could only be created by people who already knew how to make a website." (Blood, 2000) Today, sites such as blogger.com, blogspot.com or LiveJournal.com offer free opportunities for anyone to set up her own Web log. Blogs can acquire a community of readers who are able to subscribe to articles (using RSS technology), and can often add their own comments directly. Rebecca Blood, one of the most noted authorities on blogging, notes „I strongly believe in the power of weblogs to transform both writers and readers from »audience« to »public« and from »consumer« to »creator«." (ibid.)

Although Blogs are considered a medium for users writing the Web, they are usually centered around a single author. Also, the form of Blogs is fairly fixed: entries have a heading, some article text, not extending the size of a single page, and usually some links to external information. This limits the uses of a Blog to active participation in someone's reflection of the world's situation—collaboration is only possible when multiple Web log authors reflect upon one another. By contrast, *Wikis* (Leuf, & Cunningham, 2001) have been conceived as collaborative authoring spaces, and received much attention as a simplistic alternative of providing a writable section of the World Wide Web to a specific user group—largely by avoiding fixed syntax limiting the discourse within the Wiki (Obendorf, 2004).

Description

A *Wiki* is a web site where every page is editable by anybody using a normal Web browser. This is achieved by adding an edit command to every page, and by defining a simplified markup syntax that is converted to HTML by the Web server. Links are authored using WikiWords[45], words containing a second large letter, and new pages can be created

44. Apart from the academic prototypes that often were not suited for productive use, there used to be some smaller companies trying to sell annotation tools for the World Wide Web (e.g. Third Voice (Weinreich, Obendorf, & Lamersdorf, 2001)). Thus far, they seem to have had little economical success, and have quietly disappeared. Even Microsoft pulled back their annotation mechanism „Smart Tags" (Obendorf, & Weinreich, 2003) when public protesters feared a harmful concentration of annotation services—and thus, of advertisement sources (Mossberg, 2001).

45. Newer dialects often allow the definition of arbitrarily named link anchors using bracket markup. This allows external links to be integrated seamlessly within the text.

simply by mentioning them on other pages. A Wiki thus provides a very simple, and very general way of „writing the Web". It has become very successful—the oldest Wikis date back to 1995, and have been used to form e.g. the pattern movement in software engineering[46] or the Wikipedia as an alternative to traditional encyclopedias. As is noted on Cunningham's Wiki front page: „The beauty of Wiki is in the freedom, simplicity, and power it offers." (Cunningham, 2006)

Figure 87: Front page of Ward Cunningham's Wiki at http://c2.com.

In contrast to the previous examples that provide a unifying function, where simplicity and a central position in everyday work combine to trigger the use of one application for tasks it was not meant for, Wikis are intentionally designed without a specific task in mind—the understanding of what a Wiki consists of is located at a conceptual level.

The actual technology is developed for a specific use case: consisting of a bunch of Perl scripts, the initial WikiWiki software was hacked to provide a working solution for Cunningham's own needs (Leuf, & Cunningham, 2001). The general concept was taken up by others, yet interestingly, the first adopters who set up their own sites often created software by themselves; still today, new Wiki dialects continue to be developed. Although this is partly owing to the nature of the application (being server-sided it is often set up by programmers) and its first adopters, the many different extensions of the basic concept—access rights, hierarchy, versioning, features for discussions of changes—point towards a different conclusion: although the general idea is appropriate for different users' needs, often a specific extension or adaption of functionality is necessary.

In the case of email, different clients are very similar and have not almost an identical feature set, they even look like one another. Powerpoint is a single application. The design of

46. The *Portland Patterns Wiki* has through its intense discussions led to well-known publications, such as *Design Patterns* (Gamma, Helm, Johnson, & Vlissides, 1995).

Wikis, however, can be uniquely customized. Yet, they never become unrecognizable; their similarity lies within the concept of use, and in basic, core features. This demonstrates that a differentiation whether compositional minimalism applies to products or to concepts is difficult. The *composition* of Wikis defines a core feature set. Every application providing these features is considered a Wiki.

On the one hand, Wikis thus do not easily qualify as a ‚design'. On the other hand, they most closely resemble the work of the mimimalist artists that later lead to concept art—an idea is taking the place of the physical work. There is an important difference: every instantiation of this idea—every Wiki—is potentially unique.

This identifies a weakness in some current research: in many studies of the use of Wikis, no differentiation is made between Wikis on the basis of additional features; their development, on the other hand, shows that they were desirable in many contexts. Thus, they are probably important factors for the actual use of the software, possibly more important that what made different systems be considered as Wikis. Ignoring this, some studies are based on the assumption that they analyze the ‚use of Wikis' as a concept (comp. e.g. Désilets, Paquet, & Vinson, 2005). An outstanding example is the generation of Wikipedia—which is arguably more than software: it is a movement, a bureaucracy, a process, and a sociological phenomenon (Ciffolilli, 2003; Bryant, Forte, & Bruckman, 2005; Wales, 2005)—and it runs on a Wiki software.

Discussion

Wikis are an example not for a system, but for an idea. Including this example among other, more concrete products illustrates that concepts are here understood as possible product of design. For concepts, an evaluation using traditional methods of HCI must fail, as it is impossible to measure the effectiveness of „a Wiki", or the satisfaction it creates. Wikis are, however, rather successful, and it is important to understand the reasons behind their success. Minimalism is capable of explaining the fascination of Wikis: their strength lies within their universality, and the extreme ease with which they integrate with heterogeneous environments. This interpretation of compositional minimalism highlights that analysis of designs must therefore not be limited to the design of interactive systems alone, but that the organizational and social context becoming increasingly important.

Table 18: Minimal Aspects highlighted in Wiki Webs

Functional	Structural	Architectural	Compositional
Minimalism	Minimalism	Minimalism	Minimalism

A Minimal Assessment

Functional Minimalism: The original idea of the Wiki is extremely simple: making Web content editable, and providing a simplistic Way of creating new hypertext nodes and links using a naming convention. However, unlike the concept of Email, the Wiki idea has seen a diversification that has created a multitude of Wikis with different and very

specific competencies, which offer fairly sophisticated functionality, such as access control, versioning, integration of external services, addition of structural views etc.

Structural Minimalism: Seen as Hypertexts, Wikis can be extremely complex in structural terms. Understood as applications, the access structure of Wikis is fairly simple, from the ubiquitous search field to the aforementioned conventions for creating and linking pages. However, the many varieties of Wikis each add individual functionality, and access to this functionality is seldom consistent; consequently, users need to learn how to use each Wiki dialect separately—if the use extends beyond the Wiki's basic functionality.

Architectural Minimalism: Wikis support the integration with other functionality—they can be used as parts within a more complex system, e.g. as part of a content management system, and they can also form the basis for sophisticated additional functionality, e.g. as Wikipedia adds versioning and discussion functionality. The Wiki concept focuses on providing simple means to link information, and create new information units. As the Web demonstrated, this is a very universal concept.

Compositional Minimalism: The Wiki idea is an outstanding example for an idea whose use is almost universal. With only Hypertext as the more general notion of hyperlinked texts having received similar attention, Wikis are currently a class of applications that are being put to use for almost every imaginable context, from private Web sites to commercial ventures, and across all industries. This is possible as the Wiki idea does neither limit the form of content or use of a WikiWeb, nor inhibit additional functionality—it is almost universally combinable.

5.4.6 Word Processing

When computing was still in its infancy, word processing was considered not to count among the ‚serious' applications: „At first, the WP product was … limited with the idea being »those WP folks [the users] can't handle anything else«. The side effect in this KISS approach was the production of user-friendly software before it became fashionable." (Dorsey, 1983) Yet, soon word processing grew into a synonym for computer-supported work (comp. e.g. Xerox, 1983), and today has become ubiquitous. When job applications mention „software skills", word processing is included without mention. It became a basic computer skill—the one application (but for the Web and Email) that most students command (Hoffman, & Vance, 2005)—and it has been taught in school and universities since the late 1970s (Rickman, Sunkel, & Hobbs, 1979; Secrist, 1980). Word processing users range from pre-school children to the elderly, from the skilled professional writer to the first-time computer user. The success of word processing in mass markets can be retraced to the use of the WYSIWYG (what you see is what you get) principle (Press, 1993; Myers, 1998), and, on a related note, to the paper metaphor[47]. As a blank

47. The ‚paper metaphor' refers here to the ability to determine the layout (placement and direction) of text using a representation of a page, resembling the resulting output of a printer. A more verbatim interpretation of ‚page', unable to flow text to make room for insertions, did not find acceptance (Fatton, et al., 1993).

sheet of paper, a word processor can be used to produce almost any document—it places few restrictions on its users.

Description

Learning to use word processing tools has always been difficult—even when there were (by today's standards) less sophisticated applications in use; Mack et al. noted in 1983: „Computer text editors are powerful, but complex, tools. Particularly in the early stages of learning, the complexity of these tools can cause serious problems for users who are not experienced with computers." (Mack, Lewis, & Carroll, 1983) A suggestion taken up in Carroll's training wheels approach is that the many features of word processors inhibit learning (ibid.).

As the discussion of word processing in terms of *functional minimalism* demonstrated, only few features are actually used by the majority of users—even expert users. As long as features directly manipulate visible attributes of the generated text, they are easily learned. Other features, such as styles, become useful only when a certain way of working is assumed—e.g. when formatting instructions must be followed (Gadomski, & Kral, 1990). Many such advanced features are hidden in word processors; Microsoft Word is in itself hardly a minimal application—neither are its competitors, as competition is often used synonymously with feature comparison. Jensen Harris of Microsoft, however, reports that „In reality, the programs themselves weren't »bloated.« At least, the miles-long list of feature requests from customers indicated that, if anything, people expected us to do more in this space." (Harris, 2006d)

How could this seemingly paradox situation—only few used, but many requested features—stabilize? Harris explanation is that people seemingly use different functions (ibid.). Examining simpler word processing tools, such as Abiword, Mellel, Nisus Writer, or Papyrus Office, these are successfully used in very different contexts, and all have found a specific niche where they build their strength. Yet, they are unable to gather users from diverse contexts, let alone capture a majority of the market.

The creation of compositionally minimal word processing software seems to disqualify functional minimalism. Yet, the question remains whether Word was designed to minimize composition, or whether the addition of features is a silver bullet to seize new markets. The notion of *compositional minimalism* does not help much to clarify why a specific word processing software has become successful: how would one prove that an attribute of a software allows users to use it in different ways? It can, however, be used to reflect about the inverse: what do successful word processing applications miss that would hinder their use in other contexts.

Much of the discussion on the success of PowerPoint is applicable here, as well. However, while PowerPoint is (comparatively) simple in functional terms, and often used without modification, Word is a very complex application. Consequently, evidence suggests that users tend to customize their word processor—Page et al. (1996) found 92% of their 101 users had changed the appearance of their application. Tailoring activities can be triggered as general purpose software is put to use for specific tasks. In a sense, some users

then take on a designer's role (Lave, Wenger, Pea, Brown, & Heath, 1991); often, these customizations are a collaborative effort, and consequently, a systemization of tailoring (e.g. in organizations) tends to emerge (Trigg, & Bødker, 1994).

Discussion

Microsoft Word, a software that probably no one would link with minimality, is successfully used in a wide variety of contexts. By contrast, ‚simpler' word processing software is unable to cater the majority of users as it lacks the one or other feature that is absolutely necessary for a given potential customer—software with a *minimal functionality*, or a tendency towards *structural minimalism* often targets a specific market: Apple's *Pages* was developed for novice users, packages like *Nisus Writer* or *Mellel*[48] cater the professional writer's needs. There are specific niches for applications that specialize on competencies that general purpose software does not achieve; desktop publishing, for example, is still a domain that is dominated by special-purpose applications. *Minimal functionality* here often coincides with maximal adaption to specific tasks, which can hinder the software's use for other tasks.

Especially in corporate environments, communities of practice (comp. Wenger, 1999) can emerge that share tasks that are not well supported—even though the functionality exists, the interface is not capable of providing an easy access. Thus, some users take the role of translators (Mackay, 1990; Mackay, 1991), tinkerers (MacLean, Carter, Lövstr, & Moran, 1990), or local developers (Gantt, & Nardi, 1992), and tailor the software according to the community's needs.

5.4.7 Refining the Notion of Compositional Minimalism

Learning buildings demonstrate that human users will always change their surroundings, even things considered as immobile and everlasting as buildings. It is impossible to fully anticipate all these changes. It is also futile to build as if adaption had already taken place as this prevents real adaption. It might be possible to plan the construction of few restraints, allowing easier adaption. The history of the Post-It Note demonstrates that is more commonplace to build for the specific, and find wide adaption without planning.

Contrasting Powerpoint with the examples of Wiki and word processing, it seems on the one hand possible to support a wide variety of uses with a very intuitiv and simple application, while on the other hand the use of an underlying idea generates more and more specific functional requirements, disabling functionally minimal products to serve the needs of a majority of users. These products are found in specific niches, while a standard is set by the functional giant Microsoft Word.

The amount of customization Word is subjected to is both dependent on the technical profess of users, and the cost and time trade-off involved with adapting a complex application. Although Page et al. (1996) report that 92% of the 101 users in their field study customized their word processor to some degree, they do not report a customization any-

48. http://www.nisus.com, http://www.redlex.com

where as dramatic as the examples shown here; their „surprising 92% of participants" often made only small changes to the interface (e.g. a change in zoom setting, or displaying the ruler bar was considered customization, and only 4% of all users made any changes to the application's pulldown-menus. Furthermore, Microsoft found that only 2% of all sessions reported to them by their user feedback tools had a customized appearance[49]—not counting additional buttons added by add-ins or templates (Harris, 2006c). As Pipek notes, customization is difficult to analyze and support: „In the evaluations, the general usefulness of these ideas could be established, but the framing conditions (especially the subordinated nature of appropriation activities compared to »productive« work) pose serious challenges to the design." (Pipek, 2005, 88).

The design problem connected with compositional minimalism might be compared to what architects have to deal with when designing buildings: designs become finished only after they have been put in use (Brand, 1995). In a seminal paper on the subject, Henderson and Kyng (1991) described that other activities (learning, documenting, communicating) are crucial for the process of changing tool usages in the context of changing organisational requirements. It follows that compositional minimalism will often not be designed into a product, but that it is as much a quality of the development process that needs to support appropriation of tools. Pipek even demands that design projects should „maintain a social or collaborative approach to the appropriation activity they support" (Pipek, 2005, 89). A multitude of different perspectives exist that aim to enable personalization, customization, or appropriation of tools, from the purely social to the distinctly technical (comp. Galloway, Brucker-Cohen, Gaye, Goodman, & Hill, 2004).

The example of learning buildings adds two aspects to this discussion: (1) a layered design might be a useful approach for designers, (2) a design is not finished as it is delivered. The first insight is mirrored in Brenda Laurels comparison of design with script-writing—designers have to leave enough room for users to act (Laurel, 1993). Fabio Sergio, an italian professor for interaction design and former industrial designer puts it as follows: „Designers should strive to create relationships and structure but leave composition to the user." (Sergio, 2001)

It can be useful to make the unfinished nature of designs transparent to the user, as it is done in WikiWebs—and as the Web is perceived by David Weinberger: „On the Web, perfection is scary ... The imperfection of the Web isn't a temporary lapse; it's a design decision ... the designers weighed perfection against growth and creativity, and perfection lost. The Web is broken on purpose." (Weinberger, 2002, 79) As compositionally minimal design will want to invite the user into the design process, „an important aspect of design is the degree to which the object involves you in its own completion. Some work invites

49. Microsoft's numbers are based on the data of users who participated in the *Customer Experience Improvement Program*, an effort to provide quantitative data on what functionality is used in which context (Harris, 2006e). Although the CEIP collected data from millions of users, it is limited by the fact that many power users disapproved of any program that would send data back to Microsoft without their knowledge. Thus, the sample is bound to be heavily biased towards non-expert, and non-corporate users.

you into itself by not offering a finished, glossy, one-reading-only surface"—as Brian Eno noted about old buildings (Brand, 1995). The visibility of what is happening, and how the user can affect it will encourage users; this is also reflected in Matthew Chalmers notion of „seamful systems (with beautiful seams)" (Chalmers, Bell, Hall, Sherwood, & Tennent, 2004; see also Chalmers, & Galani, 2004; Bell et al., 2006). Sengers and Gaver stress that multiple interpretations must be allowed for a design (Sengers, & Gaver, 2006), the designer is no longer autocrat, he becomes an enabler for the user. For buildings, Brand however warns that „making services external"—as was done for the Centre Pompidou in Paris—is not a good idea as it immensely increases maintenance costs (Brand, 1995). Dan Hill (2002) lists three suggestions for designers: „1. Design simple inter-relating systems, which can be removed and replaced like components … [and] don't have to coexist to make useful product. 2. Open standards allow many people to add their minor creative addition to the mix—inspiring in turn subsequent developments. 3. [Provide the] Ability to generate something functional with the most limited set of instructions or components."

Figure 88: A building turned inside out—the services (heating, water) of Centre Pompidou.

Along these lines, an alternative to traditional customization support might be presented by architectural minimalism—separating the functionality and providing individual tools can aid users appropriating technology. This is not an entirely new concept—in 1996, Johnson and Nardi came to a similar conclusion when they investigated the tools used to create slide presentations. Their initial assumptions were that one could differentiate between software that is task generic, and software that is task specific, the latter type of software being better suited for creating slides, but the former being used for reasons of convenience. Instead, they found that many small tools were used in interaction—and as a consequence, they „conclude that developing application software that supports all aspects of a task well is extremely difficult and … suggest an alternative approach that may be more fruitful: providing collections of interoperable tools and services." (Johnson & Nardi, 1996)

Interfacing to existing tools is a central requirement for compositional minimalism; from the Post-It adhesive to the genericity of the Wiki Web ideal, it is a hallmark of simple and widespread ideas that they easily interface with existing—and especially with competing—technology.

5.5 Reflections on the four notions of Minimalism

The previous four sections have examined selected examples for their minimal qualities. As a first result, finding these (and more) examples was really simple, as many designs are complimented for their simplicity—it was only a matter of finding out in what respect a design could be considered minimal. The classification preceding each example indicates that this was more difficult. Most designs do not exemplify a single dimension of minimalism—in the actual designs, several, and sometimes antagonizing forces are often at work. Although this complicated the presentation in this chapter, the analysis of an individual design was much aided by the different perspectives created by the four notions of minimalism. It also makes sense to restrict the analysis of minimalism to the function, structure, architecture and composition *of the interface*—not of the underlying service layer. Although the two are often closely linked, taking the interface as a starting point allowed the analysis to take a use-centered perspective—while often, the interface is dictated by the functionality of the back-end, the implementation of the example tools is here interpreted as resulting from the interface design.

Minimalism, and its four different forms focusing on function, structure, architecture and composition have been useful tools to examine real-world designs. But the examples examined in this chapter have also helped to clarify the notions of minimalism. Before the individual refinements are summarized, an aspect of design that has been deliberately omitted from the discussion requires discussion: the aesthetics of minimalism.

5.5.1 On the Role of Aesthetics and Design

> Simplicity carried to the extreme becomes elegance.
> —Jon Franklin (1994)

Minimal design will never affect only functionality, it is bound to invoke an aesthetic impression. Whether it is a knife with a single function, or a remote control with a single button—minimal artifacts are bound to create an aesthetic experience. Gelernter stresses that simplicity combined with effective power creates a forceful sense of satisfaction, and equals beauty(1.2.1). However, while the function of an interface will influence its visual design, it does not completely determine its appearance; neither will the perception of functionality completely determine the aesthetic experience[50]. Two important aspects of

50. While it has been suggested that a usable design would be perceived as more attractive (Karvonen, 2000), experimental studies so far only found evidence that the visual appearance influences perceived usability (Tractinsky, 1997; Hassenzahl, Burmester, & Koller, 2003).

product design have so far been omitted from the discussion: the purposeful excitement of aesthetic experiences, and the visual (or superficial) design of interfaces.

In the liberal arts, theory is often used synonymously with aesthetics, and consequently, minimalism is understood as an aesthetic notion. In HCI, usability is usually defined as a measurement of effectiveness and efficiency of a tool, combined with user satisfaction (ISO9241, 1998). For much of the twentieth century, aesthetic judgement in art was based upon formal properties like color, rhythm, and harmony (Eaton, 1998). Eaton suggests that affect on the observer is a more expressive measure as aesthetic value changes according to context. Gadamer (1998) further proposes a cognitive element in aesthetics, where the observer's interaction with art is „playful", and joy comes from learning about the world and oneself. While aesthetics was traditionally only considered of indirect influence as part of the „graphic design experience" (Tullis, 1988, 378), or the „visual look" (Marcus, 1995), it is now recognized as an important factor for the perceived usability, or „goodness" of a design (Hassenzahl, 2004), and efforts are currently made to connect usability with ‚joy of use', and measure hedonistic qualities (Hassenzahl, Platz, Burmester, & Lehner, 2000), create heuristics for designing enjoyment (Malone, 1984; Monk, Hassenzahl, Blythe, & Reed, 2002), or define levels of design to induce emotions (Norman, 2004c). There are, however, few rules on how to create aesthetic designs, and no models that explain why designs are considered to be aesthetic. Heller (2005, 49) provides an explanation: while industrial design could develop rules for shapes, color, weight, type, volume and space out of long experience, interaction design is a young discipline. More importantly, „Interaction forms but one part of a whole solution; moreover, it is skeletal in nature, hidden under our more tangible and sensory details." (ibid.)

As the focus of this thesis is on designing *interactive* systems, it seems to be useful to differentiate the interaction design from the ‚superficial' visual design; in semiotical terms, this differentiation translates to the denotative and the connotative function of design (Eco, 1986). Packaging can play an important role and influence the perception of a product, although design often follows a primarily functional, and only secondarily a communicative intention. When industrial products are considered, a form of ‚minimalist' design is en vogue currently that extends beyond minimalist functionality and minimalist structure: Minimalism often evokes an aesthetic reaction, and thus influences the connotative aspect of design. Fewer functions often means fewer buttons, less visible structure equals less visual clutter. Both allow designers to extend on the created feeling of clarity. how important product design is for the success of products is illustrated by a user quote collected by Norman: „After plunking down $400 for an iPod I almost wouldn't have cared about the product after having unwrapped the packaging, it was that nice." (user statement in Norman, 2004c, 214) Here, not even the outer appearance of the tool itself, but even the disposable shell used to package it on the way to the customer became an aesthetic event. This demonstrates that it is possible to add to, or change, the perception of a tool without changing the tool itself.

‚Form follows function' is a motto often connected with the Bauhaus design movement. Bauhaus grew out of the German Werkbund, a group advocating a closer alignment of

the arts and industry. Following Walter Gropius' (1919) *Bauhaus Manifesto*, artists working within the design school made a conscious design decision to shun purely ornamental elements that detracted from an object's primary function. This decision was directed against the contemporary Arts and Crafts movement, and its celebration of elaborate ornamentation. The Bauhaus focused on designing highly practical objects, and on bringing good design to everyday life. It stressed the properties of material and experimented with reduction. This was mainly directed against ornamentation: while craftsmen would spend effort to paint flowers on a vase or cut patterns into a cupboard, straight lines and primitive forms were propagated by Bauhaus designers.

It might seem that this demonstrates design has only to communicate the function of an object, and that decoration is an evil in itself. The 1924 Bauhaus chess pieces by Josef Hartwig, however, stand as proof for the difficulty in drawing a strict line to decoration: In the tradition of Bauhaus, the chess pieces were made out of industrial material, and they were stripped of all superfluous ornamentation (Bobzin, 2004). This included getting rid of the symbolism of knights, queens, kings and pawns. Instead, the pieces' tops were formed to indicate the possible movement. The Bauhaus chess pieces were not a commercial success. Although they literally communicated the function of each piece, this was not what chess players were interested in. Chess players usually know how to move their pieces, but they like to think of them as knights, queens, kings, and pawns. In a way, the decoration of chess pieces had become part of their function.

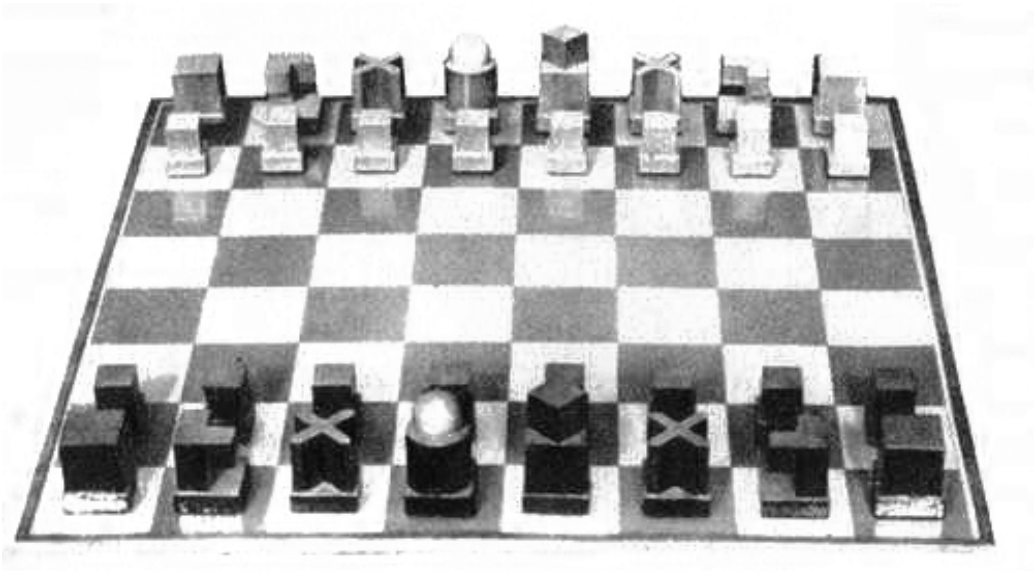


Figure 89: Josef Hartwig: Bauhaus chess set (1924).

How much aesthetic ‚packaging' is subject to Zeitgeist is illustrated by Mijksenaar and Westendorp (1999) using the example of radios: the different designs shown in Figure 90 demonstrate the differences of designs for a device playing music. Mijksenaar and Westendorp maintain that designers „try hard to make products instantly comprehensible to us, but they can't keep up with the electronics engineers and software programmers who

develop … more features"; their example shows, however, that existing functionality can be exposed (in the 1970s and 1980s) to demonstrate technical superiority, or hidden (in the 1990s) to appeal to the customers' appetite. Also, the five early models clearly show the development from a technical gadget to a piece of furniture.

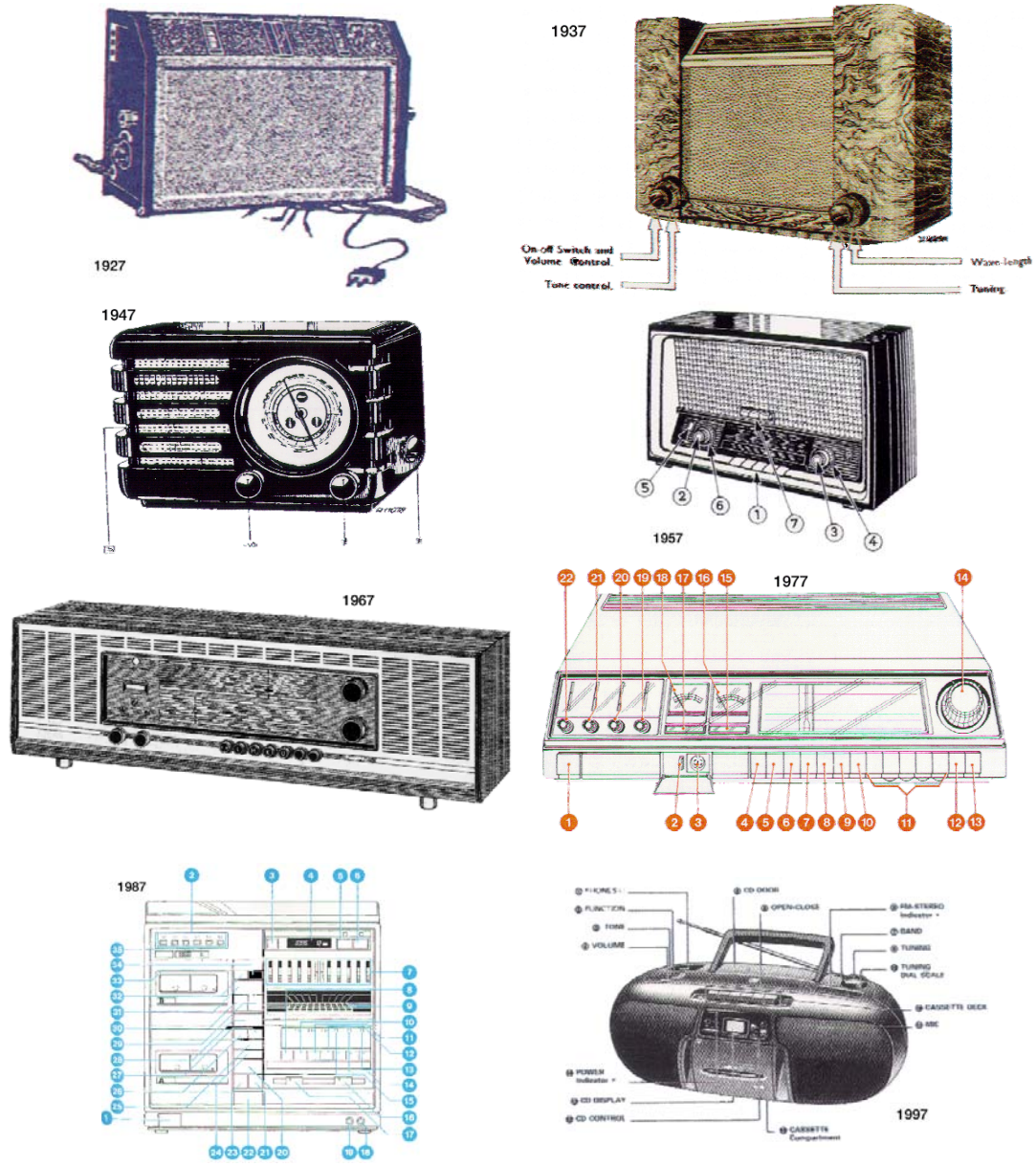


Figure 90: Popular radio designs from the 1920s to the 1980s.

The evolution of tastes for visual appearance can be verified looking at current Web designs that show a trend towards the reduction of visual clutter. As Figure 91 illustrates, navigation elements are becoming visually simplified on most sites. Although primarily a matter of taste, there is also a connection to functionality: early Web design employed many visual cues to create realistic affordances. This indicated possible activities, e.g. the

3-dimensional buttons conveyed their ‚clickability'. As use conventions, such as the placement of a navigation area, were established, the need for such communication became less pressing, and the visual appearance of interface elements was gradually simplified (Wroblewski, 2005, 3rd principle).

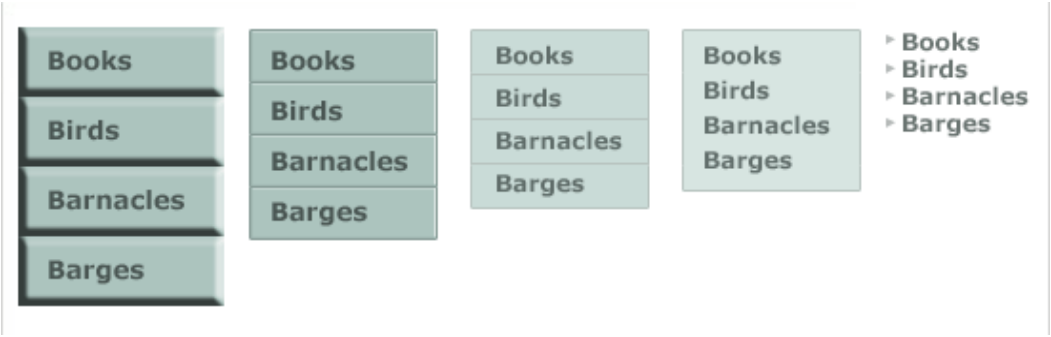


Figure 91: Navigation elements on Web sites becoming visually simpler recently. (Wroblewski, 2005)

Above evolution of graphic design could also be interpreted as a development of visual language that increasingly resembles visual design for paper media. As the display of options for interaction becomes less important, the rules of traditional design apply. Edward Tufte crafted the „1+1=3" rule (Tufte, 1986, 111) to illustrate that unnecessary visual clutter can suggest information that is not really there, e.g. two lines can cause visible interaction in for or moiré patterns or optical vibration.

The reduction of visual clutter must not necessarily lead to visually simpler designs. Tufte advocates a high data density—for him, an interface is good if it displays a high amount of information in a small space. Many examples[51] that Tufte presents in *The Visual Display of Quantitative Information* (ibid.) and *Envisioning Information* (1990) demonstrate how highly complex patterns can be detected using information visualization. A reproduction of one of his examples, Charles Joseph Minard's famous map of the russian march by Napoleon Bonaparte (1869), is shown in Figure 92; this map displays all the qualities that Tufte values in information displays: there is little visual clutter, no extra lines or borders detract attention, and all elements of the map are meaningful. The information displayed is very dense as many different information perspectives are overlaid: geographic location, troop strength, direction, and temperature are all visualized in a complex composition. Thus, this single map illustrates much of the story and the reasons behind Napoleon's march to Russia, and illustrates the great losses on the way back. Tufte demands similar qualities from information systems[52], he consequently proposes that screen space should be devoted primarily to content, shying even windows and menus (Wilchins, 1998), and preferring information-dense designs.

51. The outstanding quality of Tufte's books was reason enough to use a reference rather than a reproduction of his graphs here. If you don't already know them, please do have a look.

52. By contrast, this thesis tries to differentiate information and interaction design.

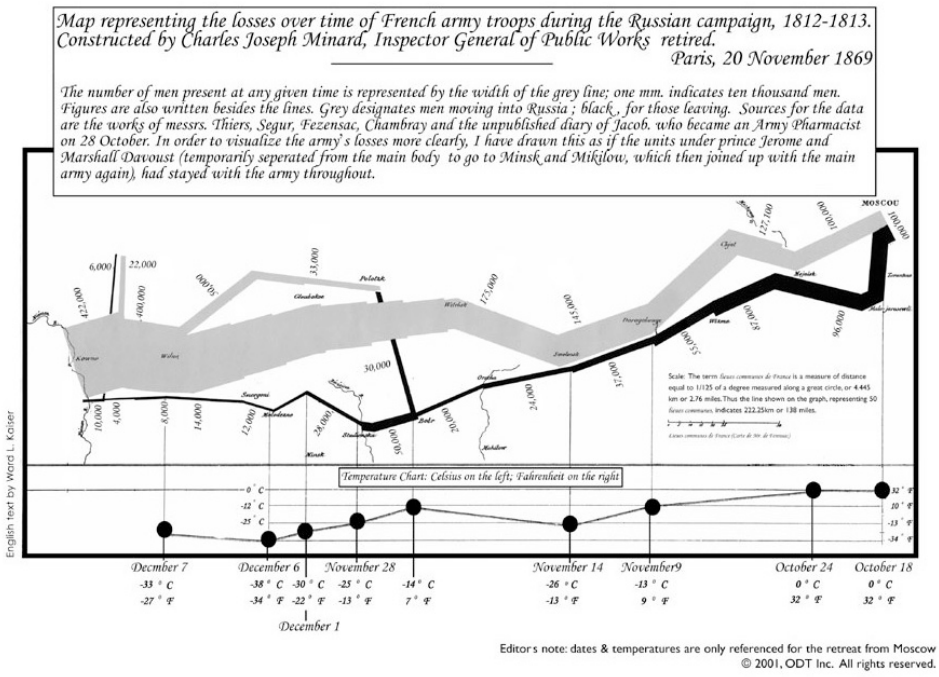


Figure 58. ***Minard s map of Napoleon s Russian campaign.***
This graphic has been translated from French to English and modified to most effectively display the temperature data.

Figure 92: Reproduction of Minard's map of Napoleon's march to Russia.

For Tufte, more *information* in the same space is preferable. He actually did count the number of links on a home page: the more links, the better the design. While this might contradict the minimalist standpoint—and Tufte in fact argues against the Google approach to access to functionality, preferring the portal functionality that e.g. Yahoo provides (compare 5.2.6)—it highlights the interaction of structural and compositional minimalism: as the example of the Minard map demonstrates, it is important for Tufte that information presents itself to interpretation; the more, the better. Pre-selection of data allows fewer interpretations, and produces less ‚rich' interaction. This is a position echoed by Don Norman, who insists "Why are Yahoo! and MSN such complex-looking places? Because their systems are easier to use. Not because they are complex, but because they simplify the life of their users by letting them see their choices on the home page: news, alternative searches, other items of interest." (Norman, 2004b).

However, he repeatedly refers to a „reduction of noise" (Tufte, 1986), and proposes that the signal-noise ratio be taken as a measure for quality[53]. The interpretation of information depends its status as „signal" or „noise", as „function" or „decoration". For those seeking information on the Web, Yahoo thus puts forward much noise. Minimal designs can thus actually display a better signal-noise ratio, and allow a richer interaction—it depends on the *context of use*. According to Löwgren (2006), aesthetic qualities, or „use qualities" of a design differ depending on *genre*. Use qualities are „properties of digital de-

53. The concept of signal-noise ratio appeals to Web designers and has been taken up as a basis for design argument by different individuals; 37signals even started a Weblog on the topic (http://37signals.com/svn/).

signs that are experienced in use and the designer is in a position to influence" (ibid., 384). In other words, they define agreeable values for users and designers, and can consequently be used to guide design—Löwgren e.g. defines what is here denoted as functional minimalism as „(technical) elegance" (ibid., 396).

As compositional minimalism is difficult to design using conventional techniques of HCI, aesthetics could be used to aid design where traditional evaluation fails—it widens the design perspective to account for appropriation of technology it (e.g. Petersen, Iversen, Krogh, & Ludvigsen, 2004; Shusterman, 1999; Bertelsen, & Pold, 2004). Bertelsen and Pold (ibid.) give the example of using notions from art history, such as „baroque" to be used for criticizing design; they aim to capture use qualities that are not focused upon by traditional usability methods. For Bertelsen, basing design upon aesthetics enables reconsideration of the dilemma between „curriculum for use" and „unanticipated use" (Light, 2005; Bertelsen, 2006). Minimalism could present a perspective for design critique based on aesthetic understanding to supplement the analysis demonstrated in this chapter. When using such an aesthetic approach, however, care must be taken to differentiate between a systematic and a folk understanding of minimalism, lest the critique be based purely on personal taste.

5.5.2 A first assessment of suitability for the Analysis of Products

Less isn't more; just enough is more.
—attributed to Milton Glaser, designer

Using a multitude of different examples, this chapter has tried to illustrate the interpretation of the four notions of minimalism, and their usefulness for assessing qualities of a design. Examples for the different aspects of minimalism were easy to find, demonstrating their discriminatory power—‚simplicity' in designs can quickly be differentiated using the four notions. The individual discussion of each example, however, explicated that a design is not simple due to adherence to a single notion of minimalism—and that it will be difficult to create good designs by following a single path to minimalism.

Functional minimalism is already a well established concept as it matches the established notion of information appliances. Perhaps surprisingly, if one applies the strict definition given here (3.3.1), most designs do not meet the criteria. Even simple technology is embedded in technological contexts that create complex interactions. Also, reduction of functionality has proven to be not enough in itself—a focus on the whole Gestalt of a design is necessary to effect increased perception of simplicity using functional reduction.

Structural minimalism is often difficult to identify as, by definition, it is not noticeable. Structurally minimal products are often very successful in interfaces that open up new markets, such as the Palm, remote controls, or a mass market cell phone. When it comes to desktop applications, structural minimalism becomes harder as applications usually burst with features that demand users access them. The example of MS Office 2007 demonstrates the superiority of a task-oriented approach: even more functionality can look like less. However, a minimal access structure must not always result in a better in-

terface: simplicity, like most elements of design, needs to be evaluated in context; if the application demands complexity, a dense presentation of information may be required[54].

Architectural minimalism is a surprisingly popular concept in more distributed environments—only for the domain of classic desktop application software, monolithic programs are the rule. This suggests on the one hand that it is possible and useful to follow this concept, and on the other hand that domains more open to the concept that are by definition both more heterogeneous and interdependent to a higher degree, and thus bound to require a more complex and partitioned infrastructure. The practical examples stressed the importance of a visible distribution[55] of responsibilities across tools.

Compositional minimalism has shown itself to be the least palpable notion. As it by definition evades application barriers, the examples discussed here have—with the exception of the Post-It and Powerpoint—more the character of services or application concepts. This indicates that flexible use is supported by a system architecture that allows the user to choose his tools and his time, and that for many specific task concepts, configuration or customization of functionality is helpful for a tool's appropriation. Some architects use incremental design, and build footpaths where people step. Designers could do the same.

5.5.3 Design Advice

To make the lessons that can be learned from the failures and successes of the different designs more accessible, guidelines for designing are extracted from the examples presented in this chapter (Table 19). Each guideline consists of three parts: a design suggestion, a list of examples relevant for the suggestion, and a note that illustrates limitations of the guideline. The individual guidelines may seem to contradict each other, e.g. prioritization of access according to frequency vs. use, as they describe alternative approaches that have been pursued to create minimal designs. Even in these cases, however, thoughtful reflection will find a way of combining different approaches if that is necessary for the design.

Functional minimalism is created reducing functionality. To still design a useful tool, it is crucial to identify the tool's core competency first, and the core functionality that is key to the design solution. There is some danger of over-reducing in the wrong places—if the resulting design feels restrictive, it is not minimal, but defective (5.1.2–5.1.4). It is therefore necessary to focus on the design as a whole. Extending a design for a second version

54. Comparisons like the one between Google and Yahoo's front page fail to include the context (5.2.6). Google is a search engine. Yahoo is a directory, and should be densely packed with information.

55. Google, the figurehead of online simplicity, made a rare acknowledgment: sometimes in product design, more is more. The distribution of functionality across a growing number of tools was part of Google's self-understanding, as Marissa Mayer noted: „Google has the functionality of a really complicated Swiss Army knife, but the home page is our way of approaching it closed. It's simple, it's elegant, you can slip it in your pocket, but it's got the great doodad when you need it. A lot of our competitors are like a Swiss Army knife open -and that can be intimidating and occasionally harmful." (Tischler & Mayer, 2005). After users of Google Video did not buy products, Mayer admitted an extra link would have helped people find the required functionality (Post-Intelligencer, 2006).

of a product must keep this focus, and at the same time find ways to optimize details of the design. Differences to the initial vision that are introduced by the addition or change of features should be reflected upon, and either the vision be changed accordingly, or the feature be put to the question. Finally, for specific designs, multiple interfaces—or multiple product lines based on a single platform—can be useful to fine-tune the amount of functionality for customers.

Structural minimalism is created as the access structures are adopted to the task. If more functionality needs to be accessible than is to be visible for practical reasons, this often involves a prioritization, and a contextualization of individual functionality. Different approaches to prioritization include assessing the frequency of use, or the user activity (5.2.2–5.2.3 vs. 5.2.1, 5.2.5). The former is most useful when use situations can be identified and analyzed easily. Both approaches are not exclusive, yet the latter carries the risk of ignoring the effects of a changing interface on the user. It is therefore most important to consider to which modalities the identified prioritization should be translated (5.2.4–5.2.5).

Architectural minimalism can present a possible way of achieving structural simplicity. It is important that the idea of modularized tools be considered on different levels of the interface—from connecting applications (5.3.2) to interaction techniques (5.3.3), and from interacting devices (5.3.4) to interconnected services (5.3.5). The modularization created by the distribution of responsibility across different tools must be made transparent to the user; fragmentation should be avoided, and the whole of the design should remain visible (5.3.7).

Compositional minimalism, finally, can be achieved by supporting appropriation as a parallel process to development: if users choose to solve tasks that a tool was not conceived for, instead of immediately trying to replace this tool, designers should first consider those design qualities that made a misuse feasible, and carefully reflect if they can be linked with the design's responsibility in a way that better supports the misuse (5.4.3, 5.4.4, 5.4.6). One way of supporting misuse is trying to devise a minimal functionality as this will often allow the user to adopt the tool for his own needs (5.4.3, 5.4.5). Another approach is to try to provide a design's functionality in form of smaller building blocks that can be interpreted differently in different use situations—an architecturally minimal system (5.3.5, 5.4.4, 5.4.5).

Table 19: Guidelines resulting from the analyzed designs

Functional Minimalism		
Identify the *core functionality* of the design. Define whether the design aims to be a simple tool.	GarageBand, CommSy, SO4K (5.1.2–5.1.4), iPod Shuffle (5.3.4)	Use whenever not targeting the geek.
Take a holistic perspective: define the responsibilities of the design *as a whole*.	GarageBand (5.1.2), CommSy (5.1.3), Palm (5.2.2), iPod (5.3.4)	Don't over-reduce, identify core competencies.
Discuss the trade-offs involved with extending functionality. Keep track of the initial vision and its changes.	CommSy (5.1.3)	This becomes more difficult as your user base grows.
Assess the feasibility of providing multiple interfaces using a single platform (optionally: let users select).	GarageBand (5.1.2), SO4K (5.1.4)	Assess whether different product lines are a feasible option.
Structural Minimalism		
Prioritize access to functionality.	all	Trade-off speed vs. balancedness
Prioritize access according to *frequency*.	Palm (5.2.2), Nokia 3310 (5.2.3), Word 97-03 (5.2.5)	Use if use situations can be identified. Task over tool.
Prioritize access according to *activity.*	Harmony RC (5.2.1), Word 2007 (5.2.5)	Use if activities can be identified. Tool over task.
Translate prioritization into appropriate modalities.	HyperScout (5.2.4), Word 2007 (5.2.5)	Use non-intrusive, contextualized media
Architectural Minimalism		
Examine your interface architecture at different levels: Building blocks can be gestures or interaction techniques, or manifest themselves as visible tools.	Automator (5.3.2), SketchUp (5.3.3), iPod (5.3.4), Web 2.0 (5.3.5)	Work at multiple levels to maximize reduction.
Modularization, or division in tools bears the danger of fragmentation—the *whole design must remain visible*.	Web 2.0 (5.3.5), Google Apps (5.3.7)	Carefully integrate all tools in one design.
Compositional Minimalism		
Designs should not be generic, but generalizable. Do not try to plan for the future, but support „misuse".	Email (5.4.3), Powerpoint (5.4.4), Word Processor (5.4.6)	Design increments, balance fit to task.
Functionally minimal designs are one approach towards design support for appropriation.	Email (5.4.3), WikiWikiWebs (5.4.5)	Use functional minimalism.
Modularize functionality in your design—form „building blocks". The resulting „use patterns" can be „interpreted" in different use situations by users.	Web 2.0 (5.3.5), Powerpoint (5.4.4), WikiWikiWebs (5.4.5)	Use architectural minimalism.

6 Minimalism in Development Processes

Almost no one seems to be able to recognize good design.
—Theodor Holm Nelson (1990, 236)

It is time to explore the role of minimalism in the creation or modification of *tools* for the design of interactive systems. As the last chapter demonstrated, the four notions of minimalism establish a minimalist standpoint that is useful for the *analysis* of designs. This chapter seeks a more constructive approach, evaluating different uses for minimalism in the design of software and highlighting possible applications of minimalism to optimize software development processes.

The chapter starts out with a direct transfer of the four notions of minimalism in form of a design game to generate design critique and suggestions for improvement. Here, the ability of minimalism to *analyze products* is used to *create a new design technique* (6.1). Minimalism is then used to *analyze development techniques* for their implicit induction of reduction; this is demonstrated on scenario techniques. Based on this analysis, an example demonstrates that minimalism can be used to select development techniques based on their reductive effects, and thus *create a development process* (6.2). Finally, minimalism is used to *analyze development processes*. The example process, taken from a real case study, used simplicity as an explicit value. From a minimalist perspective, a focus shift from functional to structural, and further to architectural and compositional minimalism is observed. Minimalism is here used as a tool to *reflect and change a genre of development processes*, and the implication of using values for developing compositionally minimal systems are discussed (6.3). A closing reflection discusses the different uses of minimalism and ends the chapter.

The Reduction of Complexity in Software Engineering and Usability Engineering

As a discipline, computer science encompasses the engineering discipline of software engineering: IEEE standard 610.12 defines software engineering as „(1) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software," and „(2) the study of approaches as in (1)." (IEEE, 1990). Balzert gives some more details about the elements

of these approaches as he defines software engineering as „goal-oriented provision and systematic application of *principles*, *methods*, and *tools* for the development and application … of complex software systems" (Balzert, 2000, 36, translation mine). According to Balzert, *principles* form the basis of engineering as they are used as a theoretical foundation; they are extracted from, and validated by experience (Balzert, 1985, 2).

Among the *201 principles of software engineering* that Davis (1995) outlines is one that reads „minimize complexity". This often cited principle is commonly interpreted in terms of by abstraction reducing the *internal* complexity—parting the inner workings of a computer program from its application as is exemplified by a saying attributed to Edsgar W. Dijkstra: „The art of programming is the art of organizing complexity." However, the IEEE standard 610.12 defines complexity as „The degree to which a system or component has a design or implementation that is difficult to understand or verify. Contrast with: simplicity." (IEEE, 1990) Complexity must thus be understood here as *cognitive complexity* (comp. Fenton, & Pfleeger, 1996, 245), or as the effort that a user must summon up to understand the workings of a computer program. It is consensus amongst many authors that there exists an *essential complexity* that cannot be avoided as it is part of the problem domain (Brooks Jr., 1987). Software engineering thus aims to reduce additional, or *accidental complexity*. As cognitive complexity is difficult to measure quantitatively, or to identify and optimize using methods and tools from engineering, structural complexity is a popular target for reduction—and thus structural simplicity a groundlaying quality of software: „Of all the principles … simplicity is the broadest and perhaps the most fundamental." (Clements, 1999) Abstraction, hierarchy and modularization are used to create a conceptual model of the application domain[56]—the latter is used as a means of eliciting different views of different users and providing a basis for informed discourse, thus enabling compromise (see e.g. Hirschheim, Klein, & Lyytinen, 1995; Mylopoulos, Chung, & Yu, 1999, 33f; Johnson & Henderson, 2002).

Analogously, *usability engineering* can be defined as a systematic, disciplined, quantifiable approach that applies *principles*, *methods* and *tools* from the usability practice to create usable and useful software systems. As software engineering, usability engineering has often been understood in technical terms (Nielsen, 1993)—as a „technical discipline for developing computer-user interfaces that can be readily comprehended, quickly learned, and reliably operated. Its replicable techniques reduce the risk of failure" (Butler, 1996); this view is often connected to psychological analysis and measurement, and usability activities are often summative, i.e. evaluate different designs to measure improvements. A different perspective focuses on designing *with* users, or *participatory design*. This school of usability engineering stresses the importance of the user as an expert, and proposes close cooperation—or co-design in the case of PD (Mogensen, Bødker, & Grønbæk,

56. A number of sub-disciplines of computer science have influenced the development of more or less formal conceptual models: Databases (ER-diagrams), AI and programming languages contributed to this research area (for an early account of the different disciplines, see Brodie, Mylopoulos, & Schmidt, 1984).

2000)—as a solution. This approach is formative, i.e. usability activity precedes and interleaves with software engineering.

The understanding of software development that is used in the following sections draws on both disciplines. It extends the focus of existing methods on functional and structural minimalism towards architectural and compositional minimalism: the last chapter introduced the four notions of minimalism as principles guiding the judgment of design. The next section builds upon this understanding of minimalism to create a design technique (6.1). The four notions of minimalism can also be used to guide the *selection of development techniques*; this is demonstrated using the example of scenario techniques, and the design of a development process pattern based on Extreme Programming (6.2). Finally, a case study is presented where the development process changed to accommodate users from new contexts, and focus changed from functional and structural minimalism to architectural and compositional minimalism: as traditional methods for participation could not be applied to the diversifying use contexts, value-based design created opportunities for designing a compositionally minimal system based on user participation techniques (6.3).

6.1 Analyzing Product Quality – Genesis of a Design Technique

In the last chapter, *functional*, *structural*, *architectural* and *compositional minimalism* have been applied to the analysis of finished designs. As the notions of minimalism demonstrated their value for design criticism, a constructive application as a design technique was manifest. The technique presented here has only approximate equivalents in HCI methodology, as unlike in expert reviews (Molich & Nielsen, 1990), the interface is not criticized in isolation, but the whole design is criticized in the tradition of *design crits* (comp. Wolf, Rode, Sussman, & Kellogg, 2006).

Design crits play an important role in *creative design activity*, which, unlike HCI methodology that traditionally either tries to identify requirements or evaluate effects of designs, tries to explore the design space through the creation of many parallel ideas and concepts, and subsequent selection. Löwgren (1995) sketched a consciously over-generalized picture as he contrasted *engineering design* that „is amenable to structured descriptions and seen as a chain of transformations from the abstract (requirements) to the concrete (resulting artifact)" with *creative design* that „is seen as a tight interplay between problem setting and problem solving … [and] is inherently unpredictable" (ibid.).

Typical design games support the expansion of the design space and focus on inspiration, on creativity and on divergent thought—the nature of design aiming at innovation is determined by divergent and expansionist motives. On the other hand, an important part of design is the convergence through reduction and refinement. Reduction requires criteria: A filter capable of capturing essential use activities is necessary, often the projected need for functionality is used as one. The *Minimal Design Game* described below tries to focus the designer's attention to this need for reduction, for thoughtful selection—and the trade-offs involved with necessary complexity. The central idea is to use the informal frame of a game to lower the barrier to reflection about design in terms of minimalism.

The game thus tries to bridge the gap between the theoretical discussion of the nature of minimalism and actual design practice; the four principles identified above can be directly applied to practical design.

Design games have a tradition in participatory design, where they are used to construct a ‚collaborative design laboratory' that makes it possible to ‚play around' with the design. Data from field studies is sometimes used to prepare material to be used in the design process (Brandt, & Messeter, 2004; Donovan, & Brereton, 2004). Exploration is encouraged by the ease of putting up ‚what if' questions; the manipulation of tangible artefacts clearly indicates the current design focus. Rules governing the game can be employed to structure the discussion, e.g. the definition of a common goal could increase the involvement of participants or introducing a turn-taking rule might force everyone into an active role – appropriate when end users are part of the design team. Design games are typically used in the beginning of a (participatory) design session and try to encourage lateral thinking processes, increase the designers' sensibility towards certain aspects of the context, or simply increase the feeling of a common vision within the group. Design games can also be applied when design is not progressing swiftly and uncommon alternatives are sought after.

The Minimal Design Game is a deliberate counter-design to typical design games trying to call out creativity or lateral thinking in the player. Here, the design game is used differently, as a form of *design crit*, and as a ‚Gedankenexperiment'. It tries to enhance analytic and synthetic qualities in the player, using a simple 2 x 2 grid of interpretations of reduction to encourage experimentation with dissection. The Minimalist Design Game as described here is a reflective game; it requires an existing design vision, and several different design ideas that might normally be minted into a single design. Playing the game should invite the designers to take a step back and look at the different aspects of a design to analyze where and how they match, ultimately resulting in a design that is not a combination created by adding different ideas, but one that follows a consistent vision.

One goal for the design of the game itself was simplicity—following the belief that the design of user interaction works best when simple materials are used that stress the prototypical character of design work, e.g. in paper prototyping (Rettig, 1994; Snyder, 2003). Also, the tangibility of physical artifacts simplifies collaboration and reduces distraction, as the medium of design is well known (Scaife, Rogers, Aldrich, & Davies, 1997)—without sacrificing realism (Virzi, Sokolov, & Karis, 1996). For involving several designers already working with paper on tables, a board game (see Figure 93) promises seamless design interaction. The playful nature of the game might also make reflection about the minimalist nature of the design artefact seem less daunting a task.

6.1.1 The Minimal Design Game

In the following, a short account of how a Minimal Design Game (comp. Obendorf, 2005) might be played is given, what is needed for preparation and what can be the end product.

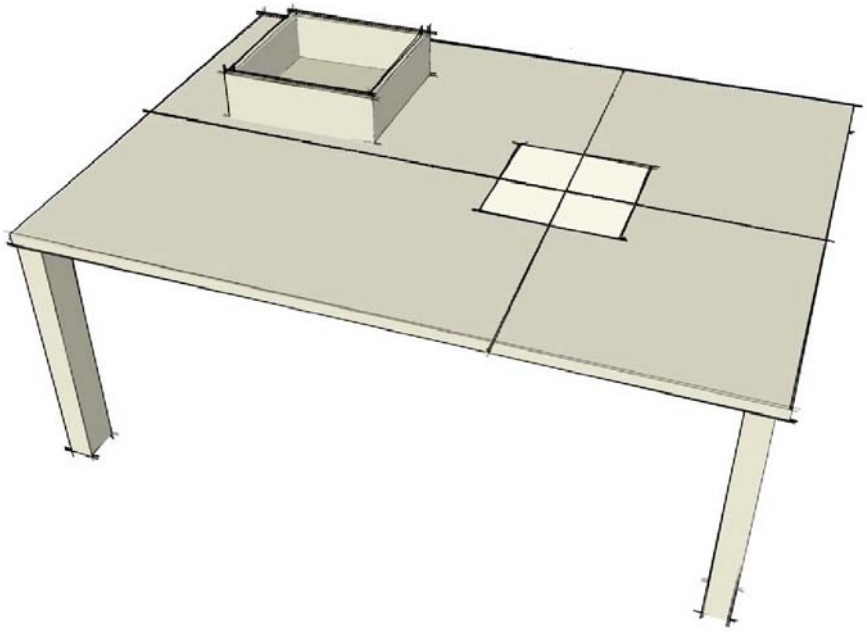


Figure 93: Table layout with playing board extended by strings (can be coiled up) and „leftovers" box.

The game proceeds in two phases: an *analysis phase* and a *synthesis phase*. In preparation, the game board is placed on a table, and then the design artefacts (sketches, prototypes, functional definitions, etc.) are placed on the table. The game logic prefers design artefacts with small granularity, so a preparation by copy and cut can ease dissection of the available material. Now, the design team can use the grid as an encouragement to think about aspects of the design that could be reduced to yield something better, where less could be more. This could mean that some functionality seems not essential, or that by omitting reference to information the interaction complexity can be reduced. It could mean that reorganizing the design around smaller, less complex building blocks could increase transparency, or that the envisioned use situations should be freed from nonessential activities.

Table 20: The spatial layout of the four aspects of minimalism on the table.

tool	1 Functional Minimalism	2 Structural Minimalism
context	4 Compositional Minimalism	3 Architectural Minimalism
	use	access

The spatial layout (compare table 20) echoes the model that represents the interactions between the different notions of minimalism (3.3.5): functional minimalism and structural minimalism focus on the design of a single tool, compositional and architectural minimalism highlight the interaction between different tools; the use aspect is dominant in functional and compositional minimalism, while structural and architectural minimalism center on access structure. To complement these abstract dimensions of minimalism, concrete design questions such as those that follow might be used to encourage thinking about the different ways of reduction that are possible:

1. What is the primary function, what is it that we really want to be able to do? What are important, but less central functions? Which functions are motivated by the context, which are motivated by technology?
2. How is that purpose of the artefact conveyed to the user? How is information presented, what are the structures used in interaction? What are the choices for the user at a given point? What model of the application will she develop?
3. Can the system be simplified by splitting it up into parts? Does the user identify and understand the simple (or already well known) parts of the design? How do they combine?
4. What is the initial use context? Did we draw any conclusions from it that limit use in other contexts? Did we introduce limitations to combine tools in our application, and if, why?

While it is not uncommon for these questions to be asked in product design, the combination and context of their asking is new—a similar relationship than that of the notions of minimalism and HCI lore (see 4.3). As the purpose of the game—reflection on the previous design with a special focus on options for reduction, finding out where ‚less is more'—was made explicit, they act together to question design aspects, probing for the essential[57].

Whenever a possible reduction is identified, the relevant design artefacts are placed on the board within the appropriate analytic dimension, and the reduction is added using post-it notes, or physically cut out of the artefact. The eliminated features, interaction methods, structural features or use situations are placed in the leftovers box. When the design team agrees that no more reduction is possible, and all aspects of the design are represented in their minimal, essential form somewhere on the grid, the first phase ends. In the second phase, the leftovers box is revisited and its contents are put in relation to the ‚core' visible on the table. This ensures that migration paths to more complex designs are held open. Whenever possible, a path to integration of the design idea is considered be-

57. Bill Buxton explained the designers' call for a „green light process", with an up-front design phase preceding software engineering (Buxton, 2003) to be motivated by the same dilemma: development often follows functional specifications, adding more and more functionality, without an understanding of the overall design vision.

fore finally burying the design idea; however, strict preference is placed on the integrity and compactness of the resulting design: what does not fit the picture will be left out. This is no strict rule, but its violations will make clear the trade-offs involved with increasing complexity.

Designing an information artefact that is in some sense minimal is already a difficult task, the design of a second version of that artifact is even more demanding. Commonly, in second versions more features are expected, resulting in more structure. If this eventual need for expansion is not considered in the initial design, it will be very difficult to find a migration path. Examples where this proved detrimental to the product's sustained success include the Palm Pilot series and its lack of affordable extensions. An assumption in the *Minimal Design Game* is that it might help to plan ahead for extensions: within a minimal design strategy, first an overall vision must be found that will incorporate both the essential design and paths for expansion. This is easier where the design artefact does not need to compete with traditional personal computers, but even there success is possible when a niche market is targeted - as perhaps the Tablet PC might prove.

6.1.2 Discussion

The *Minimal Design Game* was evaluated with a small number of students that voluntarily applied the technique to a design problem. No explicit explanation of the notions of minimalism was given before the test and only the questions reproduced above were presented as a guide for the game, as this was considered a pre-test for later use within a graduate course; the pre-test was designed to find out how much explanation would be necessary.

While this informal procedure cannot replace a more realistic evaluation, it yielded an important result: most participants immediately understood the notions of *functional* and *structural minimalism*, and had no difficulties producing advice that related to these notions. They were able to identify a significant amount of functionality, and of structure that was unnecessary for themselves; consequently, a layered approach was sought by most. Some, however, had problems interpreting *architectural* and *compositional minimalism*—although they later indicated they would have understood the terms if they had been explained to them. Although the pre-test conditions[58] are partially to blame, this underlines that architectural and compositional minimalism are not obvious and present a new perspective. This tendency was observed during the analysis of products (see 5.1-5.4), and also when minimalism was put in relation to HCI lore (4.3). Both architectural and compositional minimalism are unusual, and more difficult notions; and they both make use of terms that are already in use in computer science, and specifically in

58. A shortage of project time forbid use of the Design Game in the context it was originally conceived for, a university interaction design course—only testing with innocent by-passers kidnapped from the corridor was made in preparation, and without formal evaluation. The pre-test used an existing web site design, the Web page of the Distributed Systems Group at the Department of Informatics, University of Hamburg to simulate a realistic design setting.

software development. The misunderstandings that surfaced during the pre-test support the last argument as some participants confused software architecture with the architecture of the interface, and service composition—the developer's task—with user interpretation.

A second important result from the test setting became clear in informal post-interviews: most participants felt that a minimal solution would not necessarily be a good solution. This critical stance was welcome as it underlines a proposed value of minimalism: the use of minimalism in design critique establishes a perspective that is in itself a possible object of critique. Unlike ‚simplicity', minimalism will not be pursued blindly. Extreme models have been very successfully used as a design method (comp. Djajadiningrat, Gaver, & Fres, 2000; Dix et al., 2006) to remove unnecessary design blocks—and often, the most simple solution is the hardest to come by as it is difficult to believe that it could work being so simple.

In the pre-test setting, students had to criticize a design they were not completely familiar with, although they had previously seen it. A realistic design setting would allow design judgments of the context and make the game a more useful tool. Still, an application of the *Minimal Design Game* in a real design setting must carefully consider how much training is necessary to clearly understand the different notions of minimalism, and to enable an informed choice about which reductions to apply. It also seems to be inherently more difficult to use a game for design analysis in *architectural* and *compositional* terms, as these perspectives require a focus encompassing more than ‚just' the design artifact.

The immediate transfer of the defined notions of minimalism as values to design practice is possible only after their meaning has been negotiated and defined. Drawing on this experience, more subtle modes of using minimalism to enhance design processes are demonstrated in the following sections: first, design techniques using scenarios are analyzed and reducing effects which are obscured from the more immediate function of scenarios are identified. This allows the implementation of a design process that aims to support architectural minimalism. On a more abstract level, a real development process is analyzed from a minimalist standpoint, providing suggestions for optimization, and discovering the use of a technique to support compositional minimalism.

6.2 Analyzing Design Techniques – Genesis of a Process

From a minimal perspective, software development techniques do not only contribute towards progress in the development of a software, they can also help to reduce the design and focus on minimal qualities—or work against a minimal outcome (see 6.2.2). The application of the notions of minimalism to analyze *existing* techniques and identify those techniques suited to pursue a given minimal quality holds several benefits: (i) the practitioner can continue using her known techniques, (ii) the examined techniques are often tested and work well in real development processes, (iii) once the potential of a technique is identified, its application within a concrete development process can be adjusted to amplify its reducing function, and (iv) techniques can be selected that create designs that

tend towards a minimalist perspective without active reflection on behalf of the designer; this promises to avoid the difficulties with architectural and compositional minimalism that ‚novice minimalists' might have (compare 5.1.3 and 6.1.2).

6.2.1 Minimalism in Usability Engineering? Scenario Techniques

Scenarios play a central role in usability engineering processes—they have long been used to illustrate possible future realities (comp. Kahn, 1962) and are now widely used as tools within design for interactive systems (Rolland et al., 1998; Hertzum, 2003). Whether they assume textual (Rosson, & Carroll, 2001) or graphical form (e.g. storyboards in Beyer, & Holtzblatt, 1997), whether they are written by users (Kyng, 1995b; Kyng, 1995a; Chin, Rosson, & Carroll, 1997), by designers (Cooper, 1999), or employed as springboards (Bødker, & Christiansen, 1997) for co-design (Carroll, 2002), they are typically used to represent context as they „emphasise *some description* of the *real world*" (Rolland et al., 1998, 1, italics mine). Jarke, Bui and Carroll summarize the function of scenarios as follow: „A scenario is a description of the *world*, in a *context* and for a *purpose*, focusing on *task interaction*. It is intended as a means of *communication among stakeholders*, and to *constrain* requirements engineering from one or more viewpoints (usually *not complete*, *consistent*, and *not formal*)" (Jarke, Bui, & Carroll, 1998, italics mine). Scenarios can have a variety of functions throughout the development process (Carroll, 1995b; Kyng, 1995a). Their concreteness is often underlined, as it promises accessibility by different stakeholder groups, and thus facilitation of broad participation (e.g. Carroll, Rosson, Jr, George Chin, & Koenemann, 1998).

In usability engineering, a scenario can be used for one or more of the following purposes: (1) to create a model of the current state, (2) as a medium for exploration (Bødker, & Christiansen, 1997), (3) as a medium for cooperation of user and developer (Carroll & Rosson, 1992; Carroll, Rosson, Jr, George Chin, & Koenemann, 1998) and, finally, (4) as a (functional) specification (Rosson, 1999)[59]. The formality and exactness of scenarios varies widely with the target audience. While some of these purposes require a high level of detail, omitting detail is necessary for other tasks (e.g. for (4) to interface with UML use cases or XP user stories). Finally, the granularity of activity descriptions in scenarios varies widely, and will often be refined in consecutive series of scenarios. Although some definitions of scenarios are very broad, and include e.g. paper prototypes, or even software prototypes (comp. ibid.), the use of the term *scenario* shall in the following be limited to textual forms, possibly enhanced with graphical illustrations in the form of storyboards.

59. Rolland et al. (1998) classify scenarios according to their role in the requirements engineering process; they distinguish *descriptive, exploratory* and *explanatory* scenarios. The first two map to (1) and (2), respectively, while explanatory scenarios are used to mark human or system errors, and clarify design rationale. Explanatory scenarios thus map to both (1) and (3) here. Rolland et al. do not explicitly examine the function of scenarios as representations of a design specification—in this article, their discussion is limited to requirements engineering and stops short of the design activity.

Scenarios & Notions of Minimalism

> Drama is life with the dull bits left out.
> — attributed to Alfred Hitchcock

Scenarios are commonly used to bind a design to its context, and as contextualized designs are usually considered superior, this alone is sufficient to justify their use. Scenarios, however, have other functions that are less widely discussed. Minimalism focuses on the functionality, structure, architecture and composition of interaction design, and so the perspective taken here is not how context is represented, but how the mechanics of scenario use—both in form and contextual representation—induce reduction in designs.

Some of the uses for scenarios—regardless of the form of scenario chosen—are connected to the different notions of minimalism: (1) When scenarios are used to represent context and model the current state, they create a need for focus and conciseness—only important use practices can be included as scenarios must not exceed a certain size and their number must be kept small if they shall still be manageable. (2) In contrast to open exploration, scenario-based exploration is always (re)connected to the context; the need for grounding design ideas in real-world tasks limits the design space. (3) Scenarios force users and developers to base their argumentation in the task domain—instead of arguing about features, real world tasks are subject to design. This has a strong focusing effect on communication and co-design. (4) Again owing to the need for conciseness, scenarios as specifications for functionality are inherently incomplete. They focus on a set of core functions that are central to the developed design.

To summarize, the hypothesis examined here is that *scenarios considerably reduce the number of design options*, that they act as a *filter for design*. Whether they are used to represent activities from the application domain, present design solutions, facilitate communication, or guide the implementation, scenarios focus the design activity on an area of core functionality.

This is not due to a limited descriptiveness concerning technical innovations—everything can be imagined, while e.g. rapid prototyping tools are based on existing widget sets and command only limited functionality. Rather, the limitations of the scenario format, being textual, sequential and finite, limit the description of user interaction, effectively minimizing the described interaction. Furthermore, the strong emphasis on drawing from real world tasks and their requirements introduces a perspective that focuses on the provision of solutions for existing problems. Finally, the use of scenarios in the process as a catalyst to generate a common vision among different stakeholders limits the number of manageable scenarios, and thus amplifies their focusing role.

The representation of design thoughts in the form of scenarios considerably reduces the number of design options; designs created using scenarios tend towards *functional minimalism*, if the functionality of a design is put to the question whether it can be justified within the chosen use context. Also, the form of scenarios presents a limit to feature richness—features must be represented in a very limited number of scenarios, each of which needs to be understandable and concise. The limitation of the number of scenarios that

can be handled effectively concentrates attention to a limited number of use descriptions—and thus makes scenarios a tool to understand central aspects of the application context; the ability to combine different degrees of abstraction makes it possible to still represent the scope of development.

Scenarios can be used to focus on a single, specific use context. When used in this manner, they allow the designers to adapt the access structure to this context, creating some degree of *structural minimalism*. The abundance of more formal methods—from use cases to UML ‚scenarios'—demonstrates, however, that more detailed accounts of user activities are necessary to successfully adapt access structures. These more formal methods can be organized more easily; thus, more descriptions of user activities and system responses can be created. They lack, however, the overview effect of scenarios, and with their potentially high number do no longer prevent a loss of focus.

Scenario-Based Design

The discussion of different types of scenarios here follows their description in the literature, and draws on the experience from two courses held on Scenario-Based Design (SBD)[60] (Carroll, 1995a) held at Hamburg University. Mary Beth Rosson and John Carroll (2001) base a complete usability engineering process solely on scenarios. In several successive design activities, different types of scenarios are used to develop the initial system vision into a prototypical implementation of the interactive system. To illustrate the different stages in this process, the four types of scenarios used are first described here[61].

1. The first type of scenarios used, *problem scenarios*, tells the story of current practice, carefully tailored to reveal aspects of the stakeholders and their activities that influence design. In spite of their name, problem scenarios do not talk about current problems; their function is to ground the design in current practice in the „problem domain" (ibid., 64f). 2. The designer first introduces ideas about new functionality, new ways of thinking about users' needs and how to meet them in *activity scenarios*. Their function is to describe the future use of the design artifact, thus binding the design to concrete tasks. A more concise *vision statement* complements the scenarios, providing a more general representation of the motivation for design. 3. When the activities have been refined and confirmed upon, *information design scenarios* are used to specify representations of a task's objects and actions that will help users perceive, interpret, and make sense of what is happening. 4. Finally, *interaction design scenarios* specify the mechanisms for accessing and manipulating the task information and activities. Here, concrete interaction techniques

60. In this approach to usability engineering, scenarios are employed in different stages of the design process to describe both the current and envisioned work context, facilitate communication between user and designer, and serve as a medium for exploration. Although this focus on scenarios as a medium might seem academic, the design process illustrates the different uses that scenarios can have in real processes.

61. The documentation of design rationale in form of ‚claims', lists of trade-offs implied by design decisions, is omitted here for brevity, although it forms an important part of the SBD process (ibid., 72).

used to manipulate information are specified, resulting in a detailed description that can easily be turned into a first prototype.

The order of scenario types demonstrates the perspective taken by scenario-based design as a methodology—design is based on current practice, and re-design first targets user activities. While other processes, such as *Contextual Design* (Beyer, & Holtzblatt, 1997) do not suggest a sequence of information and interaction design[62], SBD bases the choice of interaction techniques on the choice of an information model, and thus also on the resulting data model. Constraints in SBD are passed from scenario to scenario, and the later, much more detailed design steps are consequently linked to the basic assumptions defined in the initial system vision and the problem and activity scenarios.

Each type of scenario introduces a new layer of constraints, the whole process is a form of iterative refinement. While this allows swift progress, fundamental changes in designs create the need to repeat several design steps and update the different types of scenarios.

From a minimal perspective, SBD focuses first on defining a core functionality, and later on devising an appropriate access structure. Problem and activity scenarios in conjunction describe the difference a design will make for work practice, and narrowly define the exact role it will play within the work context. User activities are described in concrete steps, and the design is embedded as a tool in the users' tasks. This not only forces designers to closely observe the necessities within the application domain, but it also facilitates the elaboration of essential functionality through communication with users. As Carroll et al. (Carroll, Rosson, Jr, George Chin, & Koenemann, 1998) put it, the focus on „human activity promotes focused reflection on the usefulness and usability of an envisioned design intervention". This shifts the focus of design in an early phase of development to the consideration of use, and of the context of use—and „mitigates the temptations and distortions of technology-driven development." (ibid.)

One of the regular elements of teaching HCI at Hamburg University is a usability engineering course. Two of these courses that used *Scenario-Based Design* (Rosson, & Carroll, 2001) as a basis form the empirical basis for the following experiences.

Experiences with Scenario-Based Design in Teaching

Describing the state-of-the-art in *problem scenarios* before doing any actual design forces the designer to step back and examine the design problem without immediately seeking answers (Carroll, 2002). Although this observation will always be influenced by the motivation for the design projects[63], this introduces a chance for reflection on real and essential user needs. According to the instructors' observations, this type of scenario was the most difficult to write for the students. They had chosen a project of their own, they had

62. Beyer and Holtzblatt instead introduce several models that describe the requirements at different levels and are consequently processed to create a non-sequential User Environment Design (UED).

63. Rosson & Carroll themselves give an example as they demonstrate the development of a virtual science fair driven by various Web technologies and the virtual community metaphor (comp. also Rosson, Carroll, & Rodi, 2004).

bright and useful ideas—and now they were forced to look at boring current procedures. It was also the most unsuccessful type of scenario with regards to capturing the important moments of interaction, and the scenario that was referenced the least in later discussions. This can only partly be ascribed to the students inexperience: without a concrete object of focus, such as the designed system, observations of the work practice are almost bound to both be incomplete and contain superfluous elements.

Nonetheless, *problem scenarios* were an important base for later design, especially as they worked as documentation of the underlying design assumptions and the initial focus of each project. Although the scenario itself was not revised, changes in the later scenarios show that the design focus shifted in all projects, with some changing course completely.

Profile

Julia is 27 years old and studies geography with a focus on latin america (LAST). Her mother tongue is german, and in school she learned English for nine, French for seven, and Spanish for two years. Julia reads much – even when she is not studying, and she‘s quite adventurous. She often goes to the movies, and prefers to watch them in the original language.

After her school exams, she travelled quite a bit through Europe with some friends before she enrolled in University. Currently, she studies in Mexico for half a year. To refresh her Spanish knowledge from school, she went to a two-week crash course during the summer holidays.

Problem Scenario

Julia is in Mexico since a month, and she has already settled in a bit. The first weeks were strenuous, but now she‘s been able to sort things out, knows her way around at the University, and has met some nice fellow students. The first language barriers are mastered, but she still has her dictionary with her all the time and uses it often.

Today, she meets with José and Patricia at Maria‘s place. They want to prepare a presentation. Julia feels that too much is asked of her, as she‘s never held a talk in Spanish before, and worries about having to concentrate too much on conversation. As Maria has prepared something to eat, they sit together, eat and talk. They plan to go to the beach together on the week-end, and Julia asks where they can rent a car for cheap. She tries to explain that she‘s got a Lufthansa Miles & More card, and that they could perhaps use that to rent a car. The others don‘t immediately understand what she means, so that Julia asks them to wait a moment. She looks up the German term „Vollkasko“ [comprehensive collision coverage] in her dictionary that she retrieves from her hand bag.

Figure 94: Problem Scenario created in the 2003 course, 1st version (translation mine).

Vision Statement and Activity Scenarios

The *vision statement* and the *activity scenarios* are tools to describe the general idea behind the design product and the way it will be used in context. It is important that this vision and the described activities are taken as guiding ideas, and not viewed as unalterable and immobile. If scenarios are not only used as vehicles to document design, but as a medium for development, repetitive refinement of the general ideas will take place when the limits of the existing concepts surface.

Activity Scenario

Julia is taking the sky train in Mexico city, on way to her study group. She wants to buy some food for dinner. In case that she only finds a small store with a clerk behind the counter, she looks up what „Ich nehme –" [I take …] means in Spanish. She takes out her Palm, ‚opens' the dictionary, and enters the term „N-E-H-M-E-N" [take]. The device provides several translations: „coger" and „tomar", as well as some phrases like „ernst nehmen" [take seriously] or „Platz nehmen" [take a seat] in Spanish. Julia reads that there are synonyms for this word that have a regionally differing meaning. Mobidic [the project's name] provides the hint that „coger" means something different in south america than in spain.

With that knowledge, she pockets the palm, exits the train and asks for food in the next grocery store.

After she arrived at Sergio's, both of them study their papers. She prepares a talk about German traditions and customs, but she can't remember the Spanish word for „Schwarzwälder Kirschtorte" [black forest gateau]. She retrieves the Palm from her pocket, and activates the device that has gone to sleep mode automatically. At the bottom of the display, the last translation is visible. She enters the new term, and finally gets a translation for Schwarzwälder Kirschtorte.

Later, Sergio's friends join them for dinner. As they eat, the young people plan a drive to the sea during the week-end. Julia owns a Miles & More card, and wants to explain that she gets a 10-percent discount world-wide, including comprehensive collision coverage. Unfortunately, she's missing the key term „Vollkaskoversicherung" [comprehensive collision coverage]. When she enters it in her Palm, the new term is diplayed above the previously entered terms, and provides the right translation.

How fortunate, she says to herself, that I always carry this precious!

Figure 95: Example for an activity scenario, also from the 2003 course (translation mine).

In our experience, the vision statement was useful as a motto that would help focus on the original design focus while the activity scenarios were used intensively to experiment with use situations (as a simulation tool) and to document design decisions. They were also the type of scenario that was most often refined, as the students discovered during the information design stage that certain features would interfere with one another, and thus a concentration on key features would be needed. This was reflected in the activity scenarios, and often they became more logical in doing so. While they initially contained a number of the designer's preformed concepts, they turned into more accurate descriptions of the future use of the software artifact.

Information Scenarios

Scenarios create the greatest minimizing effect during the layout of the information space for the activities defined in earlier scenarios. Deciding which information should be accessible at any given point within the activities defined in the previous step results in a reduction of projected features. The need to conceive of concrete examples for information forces the designer to reconsider not only physical restrictions (e.g. screen size, availability), but also the relationship between the activity and the information is put to the test – often resulting in a restriction to immediately relevant information. This effect is amplified by graphical prototyping.

Activity Scenario

Julia is taking the sky train in Mexico city, on way to Sergio, with whom she is going to prepare a presentation. She wants to buy some food for dinner. As she wants to prepare a typical German dinner, she needs some potatoes. As she prepares small stores to supermarkets, she just briefly wants to check what „Ich nehme –“ [I take …] means in Spanish. She takes out her Palm, ‚opens‘ the dictionary, and begins to enter the term „N-E-H-M-E-N“ [take]. In the entry field, she can read what she has entered so far to correct eventual mistakes. In the rest of the screen, the dictionary lists words starting with the entered letters. Julia chooses „nehmen“, and different translations appear on the screen, among them „coger“ and „tomar“. She selects the former, and is presented additional grammar information, and some phrases like „ernst nehmen“ [take seriously] or „Platz nehmen“ [take a seat] in Spanish. The word is also presented in phonetic transcription, but as she never needs that, she decides she should soon turn off the phonetic display. In addition to the translations, the dictionary provides the hint that „coger“ means something different in south america than in spain. With that knowledge, she pockets the palm, exits the train and asks for potatoes in the next grocery store.

After she arrived at Sergio‘s, both of them study their papers. For her talk about German traditions and customs, she can‘t remember the Spanish word for „Schwarzwälder Kirschtorte“ [black forest gateau]. She retrieves the Palm from her pocket, and activates the device that has gone to sleep mode automatically. The dictionary is still running as she can see from the entry field. She writes the new word letter for letter into the entry field. After she has entered „S-C-H-W-A-R-Z-W“, she needs to input an ‚Ä‘. She finds that the letter hasn‘t been recognized correctly, clears her input, and tries again. This time, the correct letter appears, and she is offered „Schwarzwälder Kirschtorte“ for choice. She selects the word, and the Spanish translation is displayed.

Figure 96: Information Scenario (translation mine).

Writing information scenarios was the phase where the most intense design work was done in the student projects. Many projects decided at this point what exactly they were going to do, or rather, what initially planned activities and what features were to be dropped from the project. The way planner became one for collaborative use in the car, the dictionary was to be used in situations of high pressure, first on the train, later on the phone; a health and nutrition database was dropped from the shopping planner. We found that the written information was less important than how the designer understood the textual representation. For our short project, we did not bother to continuously update the scenarios to reflect the actual state of design consensus, for larger projects this becomes a necessity and quite possible a vital problem of the scenario medium.

Interaction Scenarios

The specification of access to information leads to a further refinement of the design. As standard interaction techniques are selected and integrated into the design, there will be few groundbreaking changes. This is different when the use situation requires new ways of interaction that were not conceived of by the designers in advance. Writing interaction scenarios is in this way much like prototyping and iterative implementation.

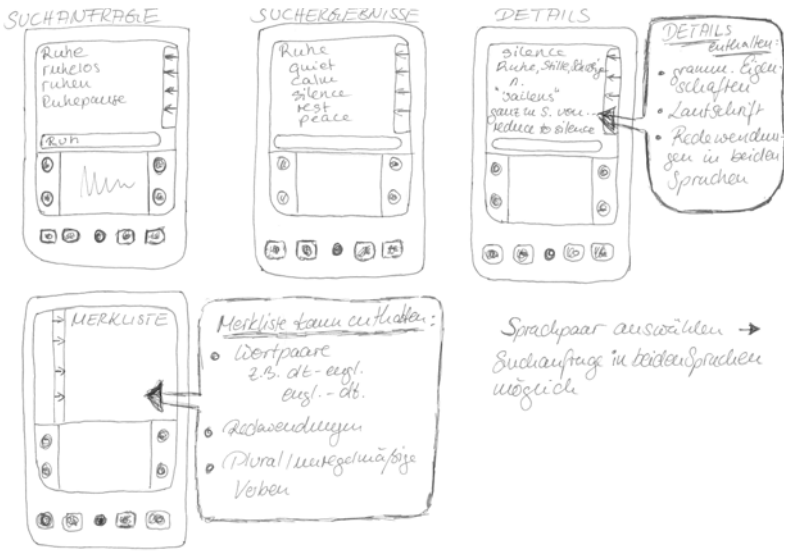


Figure 97: Storyboard accompanying the interaction scenarios (in German).

Our students soon discovered that writing stories is a very rapid method for prototyping. Actually implementing even a paper or video prototype meant taking a large step forward, as implementation details began to appear in the more graspable prototypes; scenarios were generally considered to be too soft to capture some of these problems. Also, interaction with users on a detailed level was less successful when text was used as the only medium and not complemented with a graphical representation. While no further reduction took place writing the scenarios, the implementation introduced technical restraints that in some cases also led to a simpler and more elegant design.

Difficulties arising from the Scenario Format

Scenarios are stories and the writing of stories follows rules of dramaturgy. Stories must be written to have an effect on the reader, and this limits their ability for faithful reproduction of real user needs. We found that the feeling of authors to make stories ‚interesting' was often more developed, especially as the project entered later stages and the stories were well known to all participants.

Managing the scenario life cycle—revisions of scenarios—also turned out to be a difficult endeavor. While in principle all changes were documented in the stories, it was occasionally only with computer help that new versions and their changes could be discovered. The problem was both one of storage and access, and of simple handling—which version is referred to in discussion became a source of misunderstanding. The delta between two versions is difficult to communicate and must be marked as such.

We also experienced the danger that developers tends to write ‚complete', closed stories in scenarios, pretending to have solutions applicable to a broader context and leaving open less questions for collaborative design. The experience with minimalist literature (2.3.1) suggests that a reduced description that abandons detail purposefully could generate more room for thought.

It is claimed that scenarios enable communication between users and developers on equal footing. While there are less technical hurdles to overcome, communication can still fail. A story that catches the attention but leaves room for design thoughts will still fail to work if the reader's focus is not on tasks but on other levels of the representation. While graphical prototypes direct the attention to graphical representation, stories can through bad or brilliant writing detract from the contents. In one case, we found a scenario with some spelling errors stopped the users from reading the content – they sent back a version with corrected spelling. We thus believe that scenarios must be tailored to a suitable medium (text, comic, theatre, diagrams) to achieve compatibility with the communication culture of the users.

Discussion

Scenarios, specifically when used in iterative and cyclical development, are tools that can be very effective in eliminating high-flying ideas, and carve out the essential tasks for a design. The aspect of *functional minimalism* is strengthened by two properties of scenarios, content and form:

The content of scenarios—contextual descriptions of real, or realistic use situations—forces designers to base their design on practice instead of technical feasibility, or ‚cool' functionality. Instead of *Requirements Analysis*, scenarios further *Requirements Development*: starting off from a very limited set of functionality in a single scenario, a sophisticated system is developed through iterative and interactive extension of requirements in cooperation with users. Writing additional scenarios is a way of learning about additional requirements; Carroll et al. (1998) report: „The client's original functional requirements … were radically and continuously transformed. We have called this process requirements development—it progresses through a series of stages; through the course of these stages, qualitatively different requirements become accessible or salient; the work of prior stages is often prerequisite to the possibility of succeeding stages." This does not necessarily mean that the initial vision is changed—high-level goals often stay the same during a project. Yet, by representing the redesign of workplace practice in scenarios, new requirements are detected—„Scenarios encourage designers to envision outcomes before attempting to specify outcomes, and thereby they help to make requirements more proactive in system development." (ibid.)

The *form* of scenarios also contributes to the direction of design. Functionality that seems less important is left out with less regret when a story has to be written that needs to provide justification than when a list of features allows the addition of items at no or low initial cost. As both the reader's and the writer's attention span is limited, and long stories that include a large number of similar activities quickly tend to get boring, finding prototypical activities—and matching prototypical functionality—is a main concern in scenario design. Basing design on scenarios creates an informal and incomplete specification of the design artifact. A major benefit of good scenarios is therefore that they describe an essential design kernel that will eventually grow into the final application. The

‚patterns' described by a scenario ‚seed' the development, containing much of what will emerge by addition and combination of scenarios from the beginning.

Any story—as any formal model—is created from an individual perspective. But unlike a physical model (e.g. a paper prototype), stories will often contain a morale and thus hint at the underlying assumptions. Scenarios are therefore good tools for defining a common ground (Bødker, & Christiansen, 1997) that can be used to start off design. Consistent with the minimal focus on the whole design, scenarios can also be used to understand design ideas that extend not only beyond the current tool, but also *beyond the current project* (Bødker, & Christiansen, 2004). This becomes increasingly important as designs are embedded in a technical and social context that is ubiquitous.

In commercial projects, scenarios often lack the explicit focus on exemplary activities, yet they are still employed as a focal point for the overall design vision (Hertzum, 2003), and as a basis for concentrating the design on a core competency (Greenwood, 2002; Berkun, 2002, 9). While this conflicts with the use of scenarios as experimental tools, when they must capture not just the essential and typical, but the specific and unique (comp. Grudin, 1994), scenarios also excel at describing the context of a solution. Without the need for specific details, exemplary practices can be included in envisioned solutions. However, only those vision scenarios allow the extrapolation of details that are created with a consideration for both the whole context, and the specific functional core of a design.

While scenarios do not force the designer to create systems that are structured as a combination of simple tools, they *can* facilitate the development of such systems: Scenarios are usually short stories. The need for conciseness and clarity of scenarios favors simple solutions. When different tasks will need similar features, the resulting reuse of scenario elements becomes obvious—and the different combinations that features appear in. This can be used to group the features and form categories that. The scenarios' orientation on tasks can be used to identify these categories with „tools" that have some function with regard to the task. If scenarios are used in this manner, they work as tools towards an architectural minimalism of the resulting design.

As the analysis identified several points where scenarios and minimalism meet, it might be useful to revisit the notion of minimalism in the context of computer documentation (3.2.4). While Carroll's minimal approach to documentation was here initially dismissed as being too different from the minimalist standpoint in art and music, Carroll's use of scenarios can be assumed to be based on his experience with minimal documentation. Carroll himself notes that „Design iteration was always the process touchstone for minimalism. … The shift in our technical work toward the more complex domain of programming and software design had more strongly encouraged our interests in psychological design rationale. Building explicit theories about how designs are expected to impact situations of use, as illustrated earlier for training wheels interfaces, constrains and focuses the interpretation of formative evaluation data. This is the same hypothetico-deductive logic as in the scientific method." (Carroll, Singley, & Rosson, 1992, 254-255)

Documentation minimalism was (among others) based on the premise that „working on a realistic task provides the learner with an appropriate framework for integrating and applying learning experiences" (Carroll, 1997). This quality of realistic tasks is part of what makes scenarios so attractive; as in minimal literature, much can be said with few words, because the task and the work situation add much of the unspecified information. Documentation minimalism used a minimal amount of information to describe possible strategies for recovering from error; much as compositional minimalism focuses on the interpretation of the user, and values unanticipated „misuse" of functionality, documentation minimalism tried to remove unnecessary context information from the documentation to allow the learner to make the most use of the provided textual help while trying to solve real tasks.

6.2.2 Minimalism in Agile Development

When members from the software development group and the HCI group at Hamburg university joined to design a course on developing software, they tried to combine their skills to improve the development process (Obendorf, Finck, & Schmolitzky, 2005). The mission objective was to develop a method that would combine to create a minimal design with a minimal implementation, thus enabling high coding standards, rapid prototyping, and an innovative design in a single term project—usually that time is barely enough to create either a solid software product, or a design prototype. It was soon decided that scenarios would provide an excellent means of focusing the design functionality—the analysis using the notions of minimalism has detailed the nature of their reducing effects, and that agile methods would enhance the practical realization of the design. Both methods aim to reduce the design—in the following section, the nature of reduction in agile methods is examined before a short description of the resulting process and the collected experience is given.

Agile development processes have successfully claimed ownership to the *KISS principle*[64], agile development is often used synonymously with simpler and cleaner development processes. Simplicity in agile development processes refers first to qualities of the process, and second to technical qualities of the software, while the use quality of the designed product is often not mentioned explicitly.

This is perhaps best illustrated by *feature-driven development* (FDD), a process model that predates the Agile Manifesto (Beck et al., 2001), but has repeatedly (Coad, Lefevre, & DeLuca, 1999; Highsmith, 2002; Palmer, & Felsing, 2002) been propagated as an agile process along with SCRUM (Schwaber, & Beedle, 2001; Schwaber, 2004) and XP (Beck, 1999; Beck, & Andres, 2004).

64. The acronym KISS is often translated to ‚keep it small and simple', or the less politically correct ‚keep it simple, stupid'. The term was coined long before it was used to describe simplicity in software engineering. Example uses include the description of software functionality (Dorsey, 1983), and the formulation of slogans in marketing and articles in the press; links to Ockham's razor (1654) illustrate the KISS rule's ancestry.

FDD was developed to „enable the reliable delivery of working software in a timely manner with highly accurate and meaningful information to all key roles inside and outside a project" (Ltd., 2005, 4). The process starts with the conception of an *object model* that is „more shape than content" (ibid., 6), and the building of a *feature list* describing in detail the planned functionality of the system. Development is then planned, and the software is finally designed (in an engineering sense) and built *by feature*, iteratively updating the object model (de Luca, 2002; de Luca, 1998).

FDD places its focus on the timely delivery of contracted functionality. Its main stated benefit is a reduction of the workload of its key actor, the chief architect (ibid.). It does not explicitly mention participation of users, and although it also does not forbid participation, the dominating role of the chief architect and the focus on delivery demonstrate that FDD mainly tries to reduce the risks of fixed-price projects and optimize communication between client and development team, with users taking a secondary role.

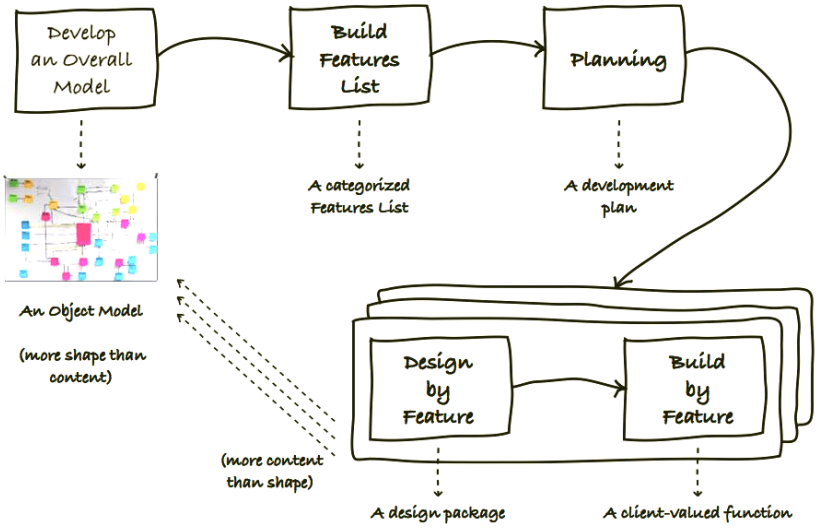


Figure 98: The FDD process. (deLuca, 2002)

No effort is made to reduce functionality of information structure of the developed software, the design is either left to the experienced chief architect, or seen as the responsibility of the client. As the software is consequently developed feature by feature, the demands on the responsible designer are tremendous as the process nowhere makes an effort to establish a general vision for development, or connect individual features to user tasks. Due to the iterative nature of (technical) design, efforts to sketch ‚the big picture' are sacrificed for an increase of the reliability of development cost planning. This carries the implicit dangers of features that only after completion turn out to be unnecessary, and that do not lead to a consistent access structure as application designers are often driven by their sense of completion; Green (1989) calls such a strategy „opportunistic design". He understands design as redesign and consequently asks for a modifiable notation—a poor fit with FDD's feature list.

Other agile processes, such as XP, also shy away from the reconception of work practice. This has several reasons; two important factors are a fear of „big up-front designs", and an economical interest in negotiation project contracts: The cost of change increases with the investment made in the conception phase. As experience has shown that changing requirements are a natural part of software development (Floyd, 1987), agile methods propagate to start with small designs and continuously change the project to fit the developing needs—this approach is described as *refactoring* (Fowler, Beck, Brant, Opdyke, & Roberts, 1999). As the reconception of work practice is a difficult and extensive process, the predictability of generated costs is limited. While in traditional, fixed-price methods, this risk is taken by the developers—hurting either the developers' profits or product quality, agile methods tend to limit the risk by accepting only concrete requests for functionality (Beck et al., 2001). As a consequence, organizational change is often not part of the developers' mandate, and no financial resources are allocated.

A Minimalist Perspective on Agile Development

From the point of minimalism, agile processes have many positive traits: technology is not developed without need (Jeffries, 2001), reducing the danger that investments in technology drive the direction of design. Further, in some agile processes such as XP, collaboration with users is an implicit part, and immediate feedback loops encourage the incremental development of requirements (Beck et al., 2001). XP also propagates an informal communication between the developers. This is manifested in the principle of collective code ownership (Beck, 1999): instead of assigning different specialist roles, and responsibilities for software modules to members of the development team, a shared understanding of all source code is desired. The second edition of *Extreme Programming: Explained* (Beck, & Andres, 2004) softens this requirement with the „core practice" called „Whole Team", acknowledging specific roles, but still demanding that „all the contributors to an XP project sit together, members of one team" (Jeffries, 2001; Jeffries, Anderson, Hendrickson, & Jeffries, 2000).

Extending this shared understanding to less formal representations of the system—such as scenarios, or other forms of documents for interface design—provides an interface to usability engineering methods. Seffah et al. (2004) described the preconditions for integrating Usability Engineering and Software Engineering as follows: „Historically, UCD has been described as the opposite of the system-driven philosophy generally used in engineering (Norman, & Draper, 1986). … This Cartesian dichotomy that decouples the UI from the remaining system and builds a barrier between engineers and psychologists is not an engineering approach. … usability specialists must think and work like engineers (Mayhew, 1999)."

6.2.3 Creating the XPnUE Process: Merging XP with Scenario Techniques

A post-graduate course that was developed together with another member from the applied and social informatics group, and members of the software engineering group at Hamburg University is presented here as an example how existing processes can be enhanced with a focus on minimalist product qualities (comp. Obendorf, Schmolitzky, &

Finck, 2006). Due to positive experiences from earlier courses teaching the XP methodology (Becker-Pechau, Breitling, Lippert, & Schmolitzky, 2003), the XP process was taken as a basis for integrating scenario techniques; our aim was to create a process pattern that would focus the design on a minimal set of tasks, and produce an original solution for a very specific design problem—a functionally and structurally minimal solution. We intended to make it possible to design a tool that would be able to integrate into existing work contexts seamlessly, and investigate the needs a full-grown product would have for interfacing with existing tools.

As agile methods reduce the effort for implementation, but do not try to reduce functionality, structure, architecture, or composition of a design, the choice of design techniques that are combined with the XP methodology is decisive for the future nature of the designed product. Hoffmann and Zerche (2005) describe the difficulties of designing a process that is a combination of Usability and Software Engineering. They choose to strictly follow the XP path and integrate *Usage-Centered Design*, a model-driven approach to user interface design that is driven by essential use cases[65] (Constantine, & Lockwood, 1999) into the XP process. Although essential use cases also try to focus on users' activities rather than atomic actions, they fail to capture the contextual richness that a scenario delivers. Consequently, no minimal aspects of design are to be expected. While XP relies heavily on ‚user stories' created in cooperation with customers (but not necessarily end users), essential use cases introduce a more formal representation; this was also proposed by Ambler (2002), who used UML system models as a means of communication with the customer. Formal methods, however, are primarily employed in communication with experienced domain experts, prohibiting co-design with real end users.

When XP relies on the customer to deliver a specification, stories play a central role in the communication of client and developer (Cohn, 2004). Scenarios build upon this communication, and add depth and context to stories. To enable the developers creating a more detailed picture of the application domain, we drew on Holtzblatt's and Beyer's description of contextual inquiry (Beyer & Holtzblatt, 1995) that tries to „make the work visible" (Suchman, 1995) as a basis for design. Scenarios were then used to represent the knowledge gathered, and to capture the use of existing tools within the context.

It is not possible, however, to add a complete usability engineering process, such as SBD, to the XP process[66]—it would reduce its agility, the primary value of the process, and hinder a reaction to change. Thus, only very lightweight techniques from usability engineering were used, and the requirements analysis was iteratively refined as a parallel activity to implementation.

---

65. Essential use cases are a generalized form of use cases as they are defined in the Unified Modeling Language (comp. Fowler, 2003), and try to capture essential interactions between the user and the system instead of individual steps.

66. Beyer et al. (2004) also had to significantly shrink their method to adapt it to agile processes.

The scenario techniques were selected based on our experiences with the SBD process (6.2.1), we explicitly chose those types of scenarios that promised to focus analysis and design activities to an *functionally minimal* perspective and to identify a *core set of functionality*—both because we intended to enable the students to deliver an innovative design solution, and because the time frame was extremely strict as it packed elements of two classes in a single course.

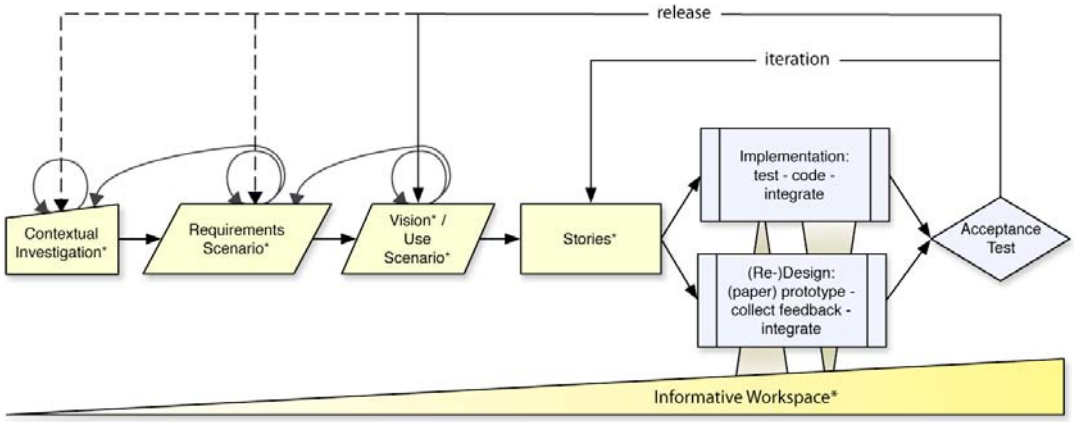


Figure 99: Process pattern developed for the XPnUE course in fall 2005. (Obendorf et al. 2006)

For the activities listed in Figure 99, only *problem scenarios* and *activity scenarios* were adapted from Scenario-Based-Engineering (Rosson, & Carroll, 2001): the *requirements scenario* rooted all development efforts in the context, it was used to identify the core functionality of the proposed system. The *use scenario* is a less strict version of Rosson and Carroll's *activity scenario*, mixing activities with sketches of partial solutions within the interface; it was used to bundle the design of functionality in a single step, directly relating it to the defined problem in the *requirements scenario*.

The XP process had to be modified, as well: our notion of stories includes storyboards as possible (and often necessary) elements. The XP tasks that are deducted from stories can lead to further contextual investigation, throw-away prototyping or the advancement of the *vision* that is used to communicate about the proposed solution with customers and end users. It was interesting to note that we could easily agree on the use of fairly standardized design techniques, such as paper prototyping or XP's test-code-integrate cycle, yet had a much harder time to define the format for representing intermediate results. As our focus on agile, iterative design forbid the continuous maintenance of e.g. scenarios as design documents or the constitution of a comprehensive archive of contextual findings, we heavily rely on informal and incomplete information stored on notes on pin boards; this „informative workspace" workes both like its related XP technique, namely takes the function of organizing and visualizing progress, and like the „affinity diagram" used in *Contextual Design* (Beyer, & Holtzblatt, 1997; Holtzblatt, Wendell, & Wood, 2004), as part of the workspace visualized the team's understanding of the problem domain and the projected solutions.

To develop the requirements scenarios, the project team needed to understand the role that an electronic calendar system played for different University personnel (secretaries and researchers) in their daily work for our departments. With this realistic setting, we experienced a large number of realistic difficulties for an academic course, ranging from dramatic schedule difficulties to feature-demanding users to capacity shortages, and some more. However, we were able to follow most of the planned activities, run through two development cycles and deliver a working prototype in the end. This would not have been possible without the vision defined during the first weeks, as it was instrumental to coordinate teams working on different parts of the project.

Many of the students told us after the course had finished that writing the requirements scenarios was one of the hardest things they were confronted with. This illustrates the different perspective obtained by the scenarios, as the students, who had some previous knowledge of XP, but no knowledge of usability engineering techniques, focused very much on individual user actions and system responses in the first version of their scenarios. These had to be redone (we misused the holidays for an additional voluntary class), and the resulting scenarios could then be used to identify the core interactions with existing software and paper calendars.

Discussion

This final demonstration of applying minimalism to the development of interactive systems was based on the analysis of scenario techniques, and the XP development methodology to inform the construction of a new development process: the analysis of reductionist effects created by extreme programming and scenario techniques allowed an informed choice—which scenario techniques should be integrated, and what aspects of the XP process needed improvement—for the synthesis of a new process pattern.

A definitive solution for fusing agile methods and usability engineering has not yet been developed either in research or practive (Birk, Kohler, & Leidermann, 2003). In contrast to previous attempts to combine XP with usability engineering methods, we were able to maintain a high agility in the process as we chose not to insert an upfront analysis phase (comp. Gundelsweiler, Memmel, & Reiterer, 2004), and still go beyond discount usability engineering (comp. Kane, 2003).

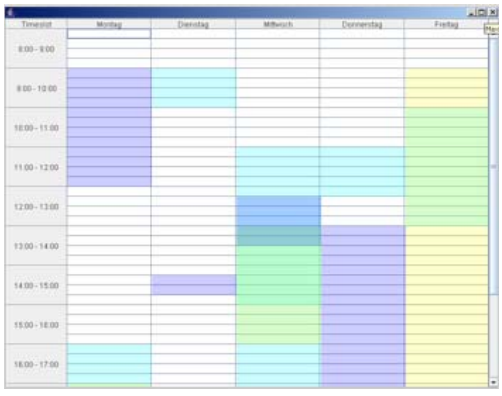


Figure 100: An early prototype of the calendar week view.

Judged by the results of the design process, the scenarios we used were very successful: the unique design that is indicated in Figure 100 is centered around the week view. Unlike other calendar tools, this view stays simple even when other people's calendars are added to the view: like planes in a geographic information system (GIS) or layers in a drawing program, additional calendars are overlaid on top of the personal calendar, supporting immediately the search for free spaces in the week where appointments could be made.

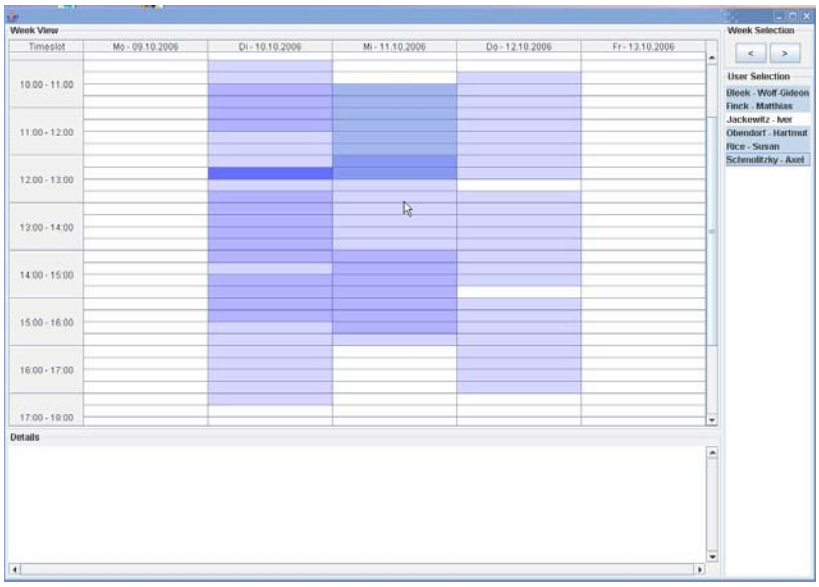


Figure 101: The final prototype, demonstrating the effect of overlaying several users' calendars.

Preliminary user tests indicated that users liked the idea; the functional prototype managed calendars for several different users, could add appointments and display the overlap of individual calendars (Figure 101); note that the initial, more colorful design was abandoned in favor of an even more reduced visual appearance—this was not initiated by the lecturers, but followed from the task focus: our students judged that if the task was only to find free calendar space, it would not help to identify who did not have time. Furthermore, experiments with adding different colors convinced our students that (de)selecting users, or creating a new interaction method, e.g. displaying the owner of an appointment on mouse-over, would be more appropriate.

Scenarios were vital for creating the design above: the *requirements scenario* identified making appointments with several, very busy persons as the key functionality, and the *use scenario* provided an solution to the problem that was incomplete and undefined in details, but addressed the whole problem situation appropriately and could be easily extended. Specifically, extending the scenario techniques with a documented design vision allowed to reflect on the whole design—although the vision itself is never complete, it allows the extrapolation of the behavior of parts of the system that are not detailed in the vision. In our experience, metaphors can be employed to make the vision more concise, yet they should not necessarily be communicated to the users—as they might not know the analogy.

6.3 Analyzing Processes – Genesis of a Process Genre

It is the simple that is difficult to make[67]
— Bertold Brecht

So far, the minimalist perspective has been successfully used to generate a technique for focusing on a minimalist design in software development, and to analyze techniques in order to select the right combination for a development process. This section demonstrates that a minimalist standpoint can be helpful to analyze not only individual techniques, but also the direction of extended software development processes. It highlights aspects of a development process selected as a case study for its evolving and dynamic nature over its existence from 1999 until today in 2006. In this analysis, two individual techniques, *intercontextual user workshops* and *commented case studies* are identified, and the general direction that development has taken is described as *value-based design*.

6.3.1 A Case Study: CommSy

The CommSy *comm*unity *sy*stem (introduced in 5.1.3) is being developed at the University of Hamburg since 1999. Aspects of its long and dynamic development history—during which it changed from a student project to a fairly large research undertaking to a commercial product were described in a book (Pape, Krause, & Oberquelle, 2004), and in several PhD theses[68]. Within the two research projects, analysis accompanied development; it was based on the researcher's own involvement in the design process, using the project as an „action case"; following Vidgen and Braa (1997), research efforts aimed to both understand the process, and initiate changes; results were documented in form of various articles. To gather additional original data, interviews with five key members of the development team were conducted. The interviews were recorded on tape, summarized in notes both during the interview and in rehearsals, transcribed for analysis and to reproduce key statements in verbatim, and important aspects were tabularized (summaries, tables, and the German transcription can be found in the appendix).

Based on the interviews, three distinct stages of development can be identified: During *The Early Years* from 1999 through 2001, CommSy was developed largely as a volunteer effort by students and faculty. From 2001 until halfway through 2004, the development setting was dominated by Wisspro—a BMBF-funded research project „Wissensprojekt »Informatiksysteme im Kontext«" (this roughly translates to „knowledge project: informatics systems in context") where many volunteers continued their work as researchers. Once Wisspro ended, and several individuals and the Virkon research project took

67. This quote, the ending lines of a poem by Bertold Brecht about the concept of communism, is often misquoted as „it is simplicity…", thus losing the context. PD's aspect of empowerment made me use it here.

68. The theses focused on the organization of software use (Pape, 2005), application service providing (Jackewitz, 2005), usability of didactical software (Janneck, 2006a), and social psychology as a basis for the design of CSCL systems (Janneck, 2006b).

over responsibilities, the development process changed and diversified, a story that is continued at the end of this section.

The Early and the Golden Years

Software development usually aims to deliver a software product made by developers to customers; this product is then used by users, who are not necessarily identical with the customers—at least three different stakeholders partake in the development process. The development of the CommSy software is an exception: During the initial stage, the development team encompassed also a majority of the user base. And although this changed quickly as the use of the software spread through word-of-mouth, two factors defining the self-conception of the CommSy development team are rooted in this stage: (1) The initial team of volunteers consisted of both faculty and students, with students taking the role of programmers, and faculty providing guidance and direct feedback, in the role of customers, users and ‚design experts'. The important role of criticism and discussion, and the understanding that developers need not necessarily program can be retraced to this setting (iv 1,4). Users were thus naturally accepted into the development team. (2) Among the early adopters were students from the humanities; they successfully used the software to support a self-organized seminar; participants were not very computer literate, yet the acceptance of the software was high and the feedback enthusiastic. Vital for this success was the high degree of learnability offered—the resulting sense of achievement informally installed the value of simplicity as one of the core qualities of the software (iv 1,4).

The development process changed, and became more institutionalized with the beginning of the Wisspro research project. As this period was both secure in financial and safe in ideological terms—the support of teaching at universities was goal of both research and development (comp. Floyd et al. in Pape, Krause, & Oberquelle, 2004, 393), this period is dubbed here *The Golden Years*[69]. During this phase, the software matured and was adapted to fit a very specific context, namely the support of courses at the university level. The design of the system was even associated with the development of corresponding teaching methods (Janneck, Krause, Pape, & Strauss, 2003; Jackewitz, Janneck, Krause, Pape, & Strauss, 2003; Janneck & Krause in Pape, Krause, & Oberquelle, 2004), although the system was also used to support more traditional forms of teaching, such as lectures or seminars (Obendorf, 2003).

As the initial development of CommSy was voluntary, and different members of the team each had their private motivations (comp. Floyd et al. in Pape, Krause, & Oberquelle, 2004, 396), the question arises whether the CommSy development process can be used to discuss methodology here. While it is certainly not true that a software development process was *applied* in the traditional sense[70], Pape argues that reflection and choice of me-

69. Floyd (ibid., 395) suggested in 2003—as financial insecurity grew closer with the end of the research project—to use this term for the initial phase of the project where progress was self-determined (yet rich in conflicts). From today's perspective, however, most interviewees see the Wisspro period as the most productive phase where team spirit, energy and task focus were high (iv 2-5).

70. As Floyd (ibid., 398f) notes, processes were considered as products in software technology from the 70s

thodology has happened „in situ" in the CommSy development (Floyd et al. in Pape, Krause, & Oberquelle, 2004, 397): as requirements and structures changed, the development process was adapted. All interviewees agree that both the initial effort and the different forms of the WISSPRO process—partly due to the conception of development itself as a *knowledge project* (comp. Pape & Rolf in ibid.), partly owing to the Hamburg school of software design—have inherited many aspects of Floyd's STEPS process. STEPS describes development on a very abstract level, leaving the concrete process design to the participants (comp. Floyd, 1993). There are, however, some peculiar specifics in the CommSy process; some of these are discussed in the following as they were quite possible crucial for the minimal qualities of the CommSy software identified in the previous chapter: the use of values provided a tool to preserve a degree of *functional simplicity* over several generations of the system, and the high degree of interaction with users, and prototyping for users ensured the adaption of well-used parts of the system for *minimal structure* in the sense of optimization of clicks, and reduction of choices.

Towards Functional Minimalism

As the development team grew, and temporarily became distributed over several locations during the beginning of the *Golden Years*, a need arose to formalize the vision underlying the project. To this end, a number of principles or *values* were codified that consequently guided the direction of development (iv 4,5). This marks a change in the communication with users: internal use was no longer dominant, and users were no longer seen as a natural part of the development team. Instead, the defined values were used with a „sense of mission" (Floyd et al. in Pape, Krause, & Oberquelle, 2004, 394) to communicate the „CommSy philosophy" (ibid.); part of the project's mission was inducing a change in the way users work with the software.

The three values that the development team arrived at after long discussions were (comp. Jackewitz, Janneck, & Pape, 2002) *simplicity*, integration into a *media mix*, and the ideal of *responsible use*. All three principles function as arguments to reduce the overall complexity of the system, mostly by providing guidelines to reduce its functionality: As CommSy was initially created for cooperative teaching in small project groups, it was assumed that no differentiation of access rights was necessary. This led to a much simpler design—access rights in file systems are inherently difficult to understand (Reeder, & Maxion, 2005)—and had a positive effect on learnability, controllability and user satisfaction. To conserve the advantage thus gained, only very simple protection mechanisms were introduced (information items that are read-only except for the author), and negotiation of *responsible system use* was taken out of the system and placed into the context—giving both the users and the system more flexibility. The integration within a *media mix* describes the assumption that CommSy should not be able to cater all needs arising within

---

onwards. Naur (1985) introduced a theory-building perspective to the controversial discussion of the use of methodologies. Consequently, for Floyd, methods have ceased to possess qualities of a product. Instead, she considers methods as resources that have to be applied to a specific project (comp. Suchman, Pea, Brown, & Heath, 1987).

even a ‚simple' university course. This acknowledges the installed base of tools already in use (e.g. email), and focuses the functionality on areas where support by existing tools is unsatisfactory and CommSy provides a real value to users. *Simplicity* was interpreted mostly in terms of functional minimalism[71]—it was often used in discussions to question the addition of a new feature in answer to a user request (iv 1-5). However, simplicity was also used as an argument to support the *addition* of a new feature—as exactly this feature would make the life of some users simpler and obliterate the need for another tool, or a workaround. For the developers, this twofold—and contradictory—use of the notion of simplicity was sometimes confusing (iv 1,4).

This confusion was caused by a seemingly paradoxical use of simplicity as the reason for two oppositional arguments. When user requests and suggestions reached the CommSy team, often by direct contact to a developer, an internal conflict was often created when the contacted developer accepted the user request as valid and useful, but other developers were not immediately convinced. In such cases, a discussion about the request was often retraced to the design values underlying CommSy. Often, both the feature advocate and his opponents were convinced that the value of simplicity was on their side. The feature advocate believed that the addition of some functionality would make tasks easier for a specific group of users; often, they would be able to do things better and easier that were already possible with the existing system—in the enhanced system, it would thus be *simpler* to accomplish these tasks. The opponents believed that a new feature would create additional complexity within the interface: the existing software was the functionally *simpler* tool. Thus, the question was whether adapting the system to a new task would be worth an increase of overall complexity (iv 1-5).

When the opposing party's criticism could not be overcome, there would often be no vote for change; but as, especially in the early years, the development process was not strictly organized, it would often happen that the advocate would prepare a prototype for the next meeting (iv 2,3,5). This ‚lazy' implementation policy led to a differentiation of feature requests in those that appeared only once, and those that surfaced again and again in discussions. Although this procedure bears the risk that the direction of development would be dominated by the more outgoing and better connected users, it served well to identify a core of features that would be important to these users.

Towards Structural Minimalism

While CommSy started as a student project without a well-defined development process, users were integrated in the development process from the beginning. About half of the

71. In contrast to the understanding of simplicity expressed in the interviews, Jackewitz et al. (2002) expand the notion of simplicity to „clear functionality…simple structure…simple layout", and „simple access". However, unlike the simplicity of functionality, the other interpretations cease to be part of recent CommSy versions: the „simple" that here denotes a unified structure was later dropped in favor of structural minimalism (comp. 6.2), and the „simple layout" is also changing in current versions with the addition of icons. Still relevant is the „simple access" by relying on Web standards, but this infrastructure-based understanding of simplicity is beyond the scope of this analysis.

members of the original development team were not developers, but interested, critical, and influential users. All members of the initial development group were interested in developing CommSy for themselves; their motivation ranged from technical fascination to practical needs for teaching. The traditional segregation of developers and users was overturned by the concerted effort. As the developers note, they „believe that the core of participation was not so much application of a technique, but a service-oriented attitude: we as CommSy developers wanted our software to be understood, and we were very keen on people liking it" (Finck, Janneck, Obendorf, & Gumm, 2006, translation mine).

Communication with users is organized in layers: documentation and personal support provide immediate help with practical questions. As the support staff consists of developers, problems are quickly identified, and often fixed within days. Additional personal consulting services are offered to key users that provide links to important user communities. Finally, workshops are offered for those users who are willing to take an active role in developing the CommSy software, its requirements, and patterns for its use. The nature of participation in the project is a result of its origins in an academic working group that initially experimented with groupware technology for their own, academic purposes. Interviewee 5, who became the project manager during the Golden Years, himself started as a participating user. This personal experience as a powerless user confronted with technological developments that could not be influenced, together with the perspective on software development promoted by Christiane Floyd (1987; 1989; 1993; Floyd, & Piepenburg, 1993) installed a tradition of participatory design for the project. While the initial development was driven to a large part by the rapid critique-development cycle of interviewees 4 and 5, the development team grew in numbers over the years, rising to a temporary high during the Golden Years. However, the number of developers was still small enough that most communication happened informally on the corridor.

In the Scandinavian tradition, the design process was considered as both organization and software development. This is echoed in the interests of the developers who tried during the Wisspro phase to not only design better software support for university courses, but to also find a methodology for courses that would fit better with the underlying concept of knowledge projects. While CommSy was primarily used in University teaching contexts, the dual interests of the Wisspro project—developing software and developing teaching methods (Jackewitz, Janneck, Krause, Pape, & Strauss, 2003; Pape, Bleek, Jackewitz, & Janneck, 2002; Pape, Janneck, & Klein, 2005)—made it possible to keep a close and informal cooperation with key users, also many of the developers themselves were vivid users of the software. Frequent *Development Workshops* were held to analyze requirements and develop designs for future developments. The form of workshops was a mix of the *future workshop* (Greenbaum, Kyng, & King, 1991) and *priority workshop* (Braa, 1995) techniques, addressing new work practice as well as resource planning. As developers, developer-users, and users partook, and every important change was discussed (iv 1), often until a consensus was reached (iv 3), a close cooperation with users was maintained. New developments were often first implemented using paper prototypes (cf. Snyder, 2003) to involve non-technical users in the development process and collect

early feedback. Using online questionnaires or semi-structured interviews, usage feedback was gathered as background rationale for future designs, reaching a large proportion of CommSy users.

Over the next years, use of CommSy spread rapidly to include other departments and universities, and – most recently – secondary schools and virtual networks of freelancers. Thus, the development team had to deal with changing user requirements and also with less and less familiar contexts of use.

An Interim Inspection

All interviewees named *simplicity* as the primary value for CommSy, and as a factor that makes the software stand out from competitors. However, that opposing parties in design arguments both used *simplicity* as a basis for their argument—with one party arguing for the inclusion of a feature, and the other against it—shows the ambiguity of this value. In terms of minimalism, a differentiation of the two conflicting arguments is possible: the tendency to ward off featurism by preventing the introduction of new functions follows the ideal of *functional minimalism*; reducing the steps necessary to accomplish a specific task by introducing or promoting a feature tries to increase *structural minimalism*. As we discussed before, striving for both types of minimalism must not necessarily create conflict observed here; while there is no guarantee that the notion of minimalism would have provided an immediate design solution, a more precise use of terms would have reduced the confusion created by the use of simplicity as a design goal.

In the case of CommSy, the introduction of features was often put off to test whether the need would show its urgency by repeated requests, and features were even removed when it became obvious that acceptance was not as expected. Due to the focused task domain, the complexity of the software could be kept comparatively small. As CommSy development was financially independent, user requests for features could (and often were) be denied; while the need for additional functionality was observed, often reflection on how to serve this need best without compromising use quality was given precedence over immediate development of the requested functionality. Direct and immediate user feedback, a high degree of prototyping and the incremental use of prototypes in production environments were vital for the identification of core features and adapt access structures. Adaptation was always dominated by the domain that developers were most acquainted with—although the use of CommSy was slowly spreading also to secondary schools, the local context at the University of Hamburg, and the contexts that were best connected with individual developers had the greatest impact on development. There sometimes was a need for compromise, but the partaking user groups were not too different. Thus, to some degree, both *functional* and *structural minimalism* could be realized within the software.

CommSy is an unusually long-term development effort, reaching from 1999 until today. As Carroll et al. (2000) observed, key users take on important roles during the development. User meeting were held in all development phases to exchange experiences, and elaborate requirements for the future development of CommSy. Users involved in these

processes also started to exchange information without the developers' intervention; typical characteristics of communities of practice (Fischer, 2004; Wenger, 1999; Wenger & Snyder, 2000) developed, such as the negotiation of meaning among the members, mutual engagement in joint enterprises, and a shared repertoire of activities, symbols, and artifacts.

Reorientation and Focus Through Values

As the last chapter demonstrated, *functional* and *structural* minimalism are notions that map to well-established goals of HCI; they are useful primarily because the support a refined and more selective analysis of what qualities distinguish a ‚simple' product. As this chapter tried to illustrate, existing HCI techniques aim to increase the usability of products by focusing the design, thus providing guidance where the functionality and structure can be adapted to better fit the task—often by reducing the design space to a core.

The observation of recent developments—whether this denotes the support for open standards, the increase of interoperability in the Web (under the labels *Semantic Web* and *Web 2.0*), or the trend from development of individual software to configuration of off-the-shelf software[72]—indicates a trend that development of successful ‚simple' products is increasingly dependent on other qualities, as well. While in the early years of software engineering, software for enterprises and distributed work was developed for a specific organization, the versatility of (CSCW) software has become an important factor for success (Grudin, 1994). This is echoed in our personal experience with CommSy, and will be retold using our analysis of the third phase of CommSy development as a case study.

After CommSy had become a large development effort during the Wisspro project, a crisis occurred when it became clear that the funding money would end soon. Several different activities resulted from the search for options to continue CommSy development: HITeC, a technology transfer company of the Department of Informatics, bundled different projects for the development and provision of CommSy, partly funded by the E-Learning Consortium Hamburg ; also, internal resources from the software technology group were mobilized to begin diverse activities, such as a migration to Java. We here concentrate on the PHP version of CommSy, and thus on one developer who started selling the provision of application services (hosting and user care), and two other members of the developments team that were able to continue their work in a new research project aiming to support virtual organizations (iv 2,3,4). These economical changes brought a shift to the direction of CommSy development: while support for the already existing use contexts had to be continued, new use contexts were added. The aim of the Virkon project was the development of appropriate software support for two different freelancer networks as a new form of work organization. Use in companies was also of in-

72. As Brooks (1987) argued famously in 1987, „software tools and environments can be bought off-the-shelf. … Any such product is cheaper to buy than to build afresh". The huge market for customization of ERP software, component-based software (Heineman, & Councill, 2001), or the widespread use of open-source tools (O'Reilly, 1999) that are adapted to cater for specific needs (e.g. Apache plus a database and scripting language for a web site) are more recent examples.

terest for the start-up, but the use of the software as an e-learning tool in secondary schools turned out to be promising, as well. As a result, the diversification of use contexts that had already been existing from the beginning quickly gained new momentum.

While the development of CommSy continued formally as an Open Source project, hosting and support services could no longer be offered free of charge. A spin-off started to provide application services (Jackewitz, 2005), and an exploration of new contexts of use began in order to find new customers. One example is the use of CommSy in secondary schools: throughout its development, CommSy had been used to a small extent by schoolteachers who had been introduced to CommSy during their studies at the University of Hamburg. Members of the development team started to intensify existing relations with schoolteachers to learn more about their requirements and use context (iv 3,4). Several workshops with teachers from different schools, and a pilot project at one school were conducted to develop use scenarios and determine how the software needed to be adapted. Teachers of different schools were brought together to exchange use experiences. New requirements were found, e.g. the naming conventions had to be adapted, and the registration process had to be simplifies for younger pupils with few or no experiences in using the Internet. Also, teacher-centered features were called for, although they were in conflict with use scenarios generated during the pilot project—and with CommSy's *design principles*.

The Virkon project proposed CommSy as a prototype that was to be continually adapted and customized due to needs of two networks of freelancers. Network members were consultants willing to exchange experiences, information, and work documents. Network contacts were also used for project acquisition. A social component of the network was the provision of mental support, and social interaction with colleagues. The members stress the importance of working together on an equal footing as well as on a voluntary basis. Financial resources are scarce. The *design principles* seemed to well match the network's proposed ideals of self-dependent work, equitable cooperation, and flat hierarchies and access rights. However, a number of new requirements regarding specific project management-related features emerged: CommSy should be used to represent the network on the Web, support acquisition processes, and, in case of successful bids, project management. Consequently, new functionality was requested, e.g. a shared calendar, and a task planner. The functionally minimal CommSy, a tool for communication among peers, was confronted with functional expectations on a number of different fields.

Despite the strong tradition of challenging new functionality in the CommSy development process, it was subjected to feature creep. Additions that in part had been rejected due to the focus on functional minimalism were added to the system; this included icons (iv 1,4), a shared calendar (iv 3), and additional categorization mechanisms (iv 3). The commercial interest of the start-up company was partly responsible—when a feature request was backed financially, resistance was seldom put up (iv 3,4), and the weakening, or leave of developers that used to be strong interceders for functional reduction (iv 1,2,4).

For development, this provided new challenges: while CommSy was used primarily at universities, the tool infrastructure was largely uniform, and while teaching practice differed from course to course, many similarities—and a common language—existed. The new use contexts differed widely in their use of tools, but also in their daily tasks and procedures, and even in the objectives for using the software. The development process thus faced a twofold challenge: it had to provide *integration with existing tools* and the *development of a community* that could partake in the development process. These two challenges can be mapped to *architectural minimalism*, and *compositional minimalism*.

Architectural Minimalism

As requirements were collected from the freelancer networks in the beginning of the Virkon project—this was done using workshops and demonstrations of the existing software and its underlying principle, collaboration on equal terms—many additional features seemed necessary to the freelancers: from previous experiences with other software they knew of functions that they wanted to be included within CommSy, as well. Most of these functions were connected to project management: One example was a calendar tool for scheduling projects, another a function to assign and manage tasks (Janneck, Finck, & Obendorf, 2006b). They also proposed to integrate email functionality within the system.

Several previous systems had been in use at the freelancers' network, and all had failed to gain a sufficient amount of acceptance. In the light of he previously listed requirements, it might seem legitimate to ask why CommSy was chosen as a system. However, the freelancers communicated that they both needed a community platform that would enable them to exchange appointments and materials—exactly what CommSy was built for[73]—and the additional features for project management in an integrated form. From a minimalist standpoint, the requirements set by these new users challenge the whole design of the system: the required functionality would make the system into something of a swiss army knife, and the dissimilarity of work procedures of universities, schools and the virtual networks posed a challenge for providing optimally structured access to existing functions. The approach taken was following the PD tradition: a careful challenge which features were those that seemed to be of most value to the users indicated the areas where CommSy was to be extended. This extension was built using „flagged" code: using conditionals, features embedded in the page templates were only accessible when the database record for the specific project room indicated it (iv 2,5). The addition of features for a specific user group would thus not increase the complexity of the software for other users—for them, there would be no perceivable difference.

The CommSy developers shied away from the option of developing and supporting several distinct products for different user groups as the available work force was limited. In

73. Interestingly, and we will return to that point, the values proposed by the CommSy team were wholly embraced, the freelancer communicated that their work practice exactly matched the underlying idea of collaboration on equal terms.

a retrospective analysis of the development process, the developers state that „the development team needed to bundle resources and tried to avoid parallel implementations that would increase the complexity of software and difficulty of administration and maintenance" (Obendorf, Finck, Janneck, & Bødker, in prep.). Furthermore, they wanted to make sure that no fragmentation of the development team and process would happen (ibid.). This is a common problem for open-source projects when the motivations and aims of individual developers differ too much from one another (Feller, & Fitzgerald, 2000, 65; Nakakoji, Yamamoto, Nishinaka, Kishida, & Ye, 2002, 82).

Thus, the development team thus chose to implement a modular, yet „closed" approach: for those users who needed functionality not provided by the base system, the system was extended. Usually, this was done by adding a new information „category". While this procedure imitated the prototype-based development process implemented from the beginning, the pre-conditions were different in this stage of development: the aim of prototypical development was no longer the functionally minimal „whole", but the fulfillment of feature requests. Reflecting on this change in motifs for development, the question whether an alternative procedure might have been preferable comes to mind. From the standpoint of architectural minimalism, the answer is clear: while the separation of functionality within the system was applied, integration with other tools might have been preferable. To the user, the web-based CommSy appears as a single tool. The choice of a category puts the user in a special mode; to limit negative impacts of modality, a high degree of consistence was aimed for. This result in severe limitations of what new functionality can be introduced as a new category.

Although users generally perceive CommSy as being easy-to-use, and are very satisfied overall (Janneck, 2006b), the longer development of the system continues, the more features are added. This poses the risk of making CommSy a complex system and harming its central value of simplicity. The evaluation of recent CommSy use in two university lectures (5.1.3) indicated that some users feel that the current system is too complex—even if it is still much simpler than competing CSCW systems such as BSCW (a detailed comparison can be found in Wolfhagen, 2006).

Furthermore, it is doubtful that the integration of new features is compatible with the initial design ideas that were made explicit in the development values (see above in this chapter). The ideal of using CommSy within a *media mix* was mainly conceived of due to the inappropriateness of replacing email as a medium for communication within a university context (iv 3). Hidden within this ideal is the idea of architectural minimalism—that other, often more mature tools are available for specific tasks, and that these tools should be used in conjunction with the CommSy system, not be replaced by it. In fact, the similarity of the initial value with architectural minimalism became more obvious as the value changed from *media mix* to *openness* recently (transl. from German „Integrierbarkeit", iv 4, Finck, & Jackewitz, 2006).

Recent developments in the Web (compare 5.3.5) suggest an architecturally minimal approach for CommSy as a Web-based application: Functional requirements, such as the

calendar and task tools for the freelancer networks, or the knowledge organization for the study.log project (Meyer & Münte-Goussar, 2005) were integrated into the core product, that consequently became more and more complex. Even if the *core* functionality was only extended hesitantly, those systems that had the sophisticated functionality enabled, offered an increasingly complex interface: the number of visible tabs, and the information density of pages for individual items increased.

An alternative approach would trade-off the high visibility of additional functionality in the existing system for decreased complexity. Existing tools, such as Microsoft Outlook or Apple iCal could be used to manage tasks and todos, or a group calendar. This approach is limited where existing tools do not expose an interface that could be used to link applications, e.g. in the case of project management. With the change of the *media mix* principle to *openness*, CommSy development is beginning to take this path: Functionality integrated during the Wisspro phase, when two groups merged the core CommSy, and a knowledge archive (iv 2,4), was made optional with CommSy 3.0. The portal functionality offered an overview of project rooms, and allowed to exchange information items between rooms, or publish public information. This introduced an additional layer into the system that was not necessary for many users, who were content to work within their closed room. As the portal was made into a separate tool, users can now decide whether to use it, or not.

Among other reasons, the integration of CommSy into the complex new university information and management system made necessary the development of interfaces to the data stored within CommSy. By providing services to other web applications (e.g. automatic generation of a project room on registration of a course), and drawing on existing services (e.g. using the commented course list to display inline information about a project room), functionality will be added for the user without introducing additional complexity to CommSy.

From the perspective of architectural minimalism, this course of action is highly desirable and should be intensified: a much more powerful—and still simple—system could be created by interconnecting the CommSy core with other dedicated applications. Rather than integrating functionality, the power of existing applications should be levied. For e.g. email, this would mean to use the user's email tool to send group mails, possibly providing mailing list functionality; this would ensure that the user had copies of all of his email actions in the usual place instead of spreading traces over different systems.

Compositional Minimalism

With the shift in economical funding, not only the objectives of development, but also the context for participation changed: Instead of facing a single, homogeneous group of users, the CommSy developers became increasingly involved with very different groups of users. This required a different approach to participatory design: Traditional PD focuses on support for a single *community of practice*[74], which may consist of very different stake-

74. *Communities of practice* were already used by John Seely Brown, „They are peers in the execution of »real

holders that share a common practice (comp. Wenger, 1999). Building and fostering this community of practice helps not only create an pool of participating users, but enables them to help themselves (this is retold in more scientific terms in Wenger, McDermott, & Snyder, 2002). Some of the specific requirements elaborated within these communities of practice are of little or no significance in other contexts. This posed a problem for the development of CommSy: On the one hand, exploring new contexts of use—and thus, new customers—was vital to the commercial interests of the spin-off company. On the other hand, the development team needed to bundle resources and tried to avoid parallel implementations that would increase the complexity of software and difficulty of administration and maintenance. Furthermore, they wanted to avoid the high risk of fragmentation of the development team and process, which is known as forking in open source projects. Also, the different requirements had to be aligned with the original design principles.

6.3.2 Towards Value-based Participation

During the development of the CommSy software, a need for catering to different contexts arose, and the methods for participation were adopted to address this need. Although it was not planned in advance, the use of *design principles* turned out to provide a good basis for discussing design when stakeholders span several very different contexts: while traditional PD methods suffice to work within a *singular* context of use, or *community of practice*, the newly formed methodology was successful in establishing a manifold *community of interest*.

Value-Based Design is an attempt to formalize the experiences with the CommSy development project. It is laid out as a form of *Participatory Design* (PD), which has often focused on the quality aspect of software development (see e.g. Bødker, Grønbæk, & Kyng, 1995): users are considered to be domain experts, the involvement of users thus yields better requirements specifications, and better design. The European labor movement following the goals of humanization and democratization of work is a second aspect behind PD (Floyd, Reisin, & Schmidt, 1989; Braa, 1996). However, PD's focus on large organizations and ‚workers' cannot easily cope with software whose use extends beyond a single context, and where boundaries between stakeholders seem more fluid (Greenbaum, Snelling, Jolly, & On', 1994). PD techniques are useful to criticize and develop existing work practice in a single community of practice with similarities in daily work practice and users' tasks. Even when different stakeholders participate in design (comp. e.g. O'Day et al., 1996; Korpela et al., 1998), they are involved in a single context.

---

work«. What holds them together is a common sense of purpose and a real need to know what each other knows. There are many communities of practice within a single company, and most people belong to more than one of them." (Brown, 1990), and made popular by Etienne Wenger who defined them as "a group of people informally bound together by shared expertise and passion for a joint enterprise." (Wenger & Snyder, 2000)

The aim of PD is in minimalist terminology the creation of a software whose fit to a single context is optimal—often, structural minimalism is sought. For multiple contexts, it is necessary to establish a common base of experiences first, and consequently develop sensitivity for different perspectives among the users to enable them to reflect on *software use as tool appropriation*. The notion of *compositional minimalism* is thus highly relevant when designing for multiple contexts.

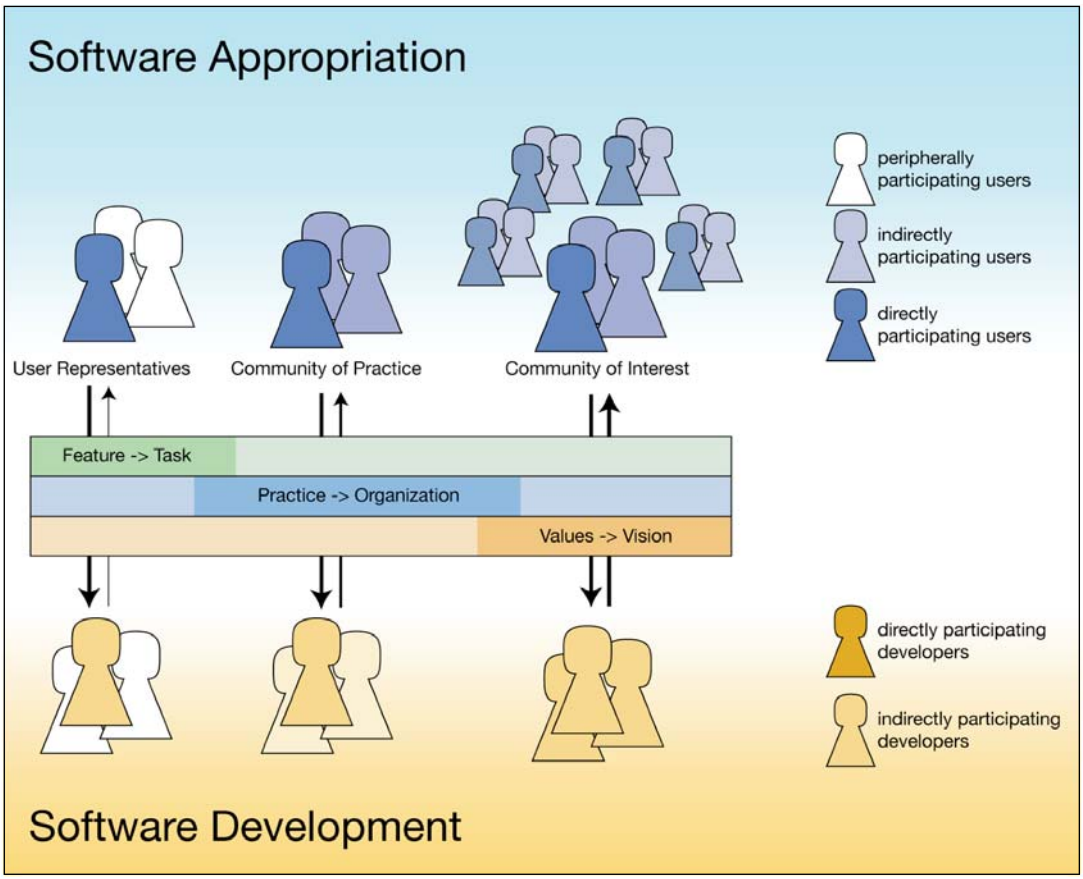


Figure 102: Extension of scope and increasingly abstract level of discussion. (Obendorf et al., in prep.)

The experiences from the CommSy case study can be generalized based on the distinction between the different levels of involvement that coexisted during the development process (see Figure 102): communication with individual user representatives, with members from a single community of practice, and with representatives from several different communities of practice that should form a community of interest.

Communication with user representatives took the form of personal one-to-one links of key users and individual members of the development team; the nature of this participation could be described as feature-based—design requests often concerned a single feature and were based on immediate task requirements. Both *communication with a single community of practice* and *communication with representatives from several communities of practice* happened in workshops, and the *commented case studies* were used to document the

discussion within the community of interest. The participation was based on organizational practice, and design principles, or a vision of use, respectively.

With the increasing dissemination of CommSy in different contexts, the importance of *compositional minimalism rose*, and the development of *design principles* for the software acquired a central role. The communication between domain experts and programmers made values useful even in the early phase of the development process, yet with the addition of different, heterogeneous contexts, the design philosophy that captures the rationale for design decisions took on another role: values were used to communicate the reason for taking one design direction and consciously ignoring other options to users. The design philosophy acquired a communication function, and changes in the direction of design became visible early as the values favored in design discussions shifted. The move towards a *design philosophy* made it possible not only to discuss the inherent design values with users, but also to develop them further and assess their validity.

Gilbert Cockton has postulated that computing, psychology and sociology have all failed to deliver value to customers, and that „our role should be to understand what is valued by a system's stakeholders and support them in delivering this value" (Cockton, 2004). The main focus of *value-based development* is exactly this negotiation of values. A shared understanding of values is necessary to create designs valuable for the user. But value-based design extends this understanding to the development of existing systems: as here, values have been established both within the existing use community, and within the development team, it is no longer possible to extract values from users, and try to cater to their needs. Instead, the negotiation of shared interests becomes a central issue for development processes.

Techniques Developed and Analyzed

To meet this new challenge of pooling the interests of different communities and the development team, new ways of bringing users from different contexts together had to be established to balance their respective needs and upcoming requirements. The goal was to establish a community of interest (O'Day et al., 1996) across the different communities of practice, united less through common practices in their respective work context, but rather through a shared interest in CommSy use. *Communities of interest*, defined as „communities of communities" (Brown & Duguid, 1991), do not share a common practice. They are instead defined by a shared interest (Fischer, 2001; Fischer, 2006). For software development, this means that if one can build and foster a community of interest (in this case, in the developed software), chances are high that a mode of communication with users can be established that allows to draw conclusions about shared values, and similarities in the requirements for software use—even if the individual work practices are dramatically different. Thus, a shared perspective, or vision, could be formed, and „a joint development without overly neglecting specific use contexts" (Obendorf, Finck, Janneck, & Bødker, in prep.) might become possible.

Building such a community of interest would allow—at least to some extent—a joint development, reducing the risk of neglecting specific use contexts. To this end, two tech-

niques were developed to create and nurse a community of interest: *intercontextual user workshops* and *commented case studies* (for further details, see Finck, Janneck, Obendorf, & Gumm, 2006).

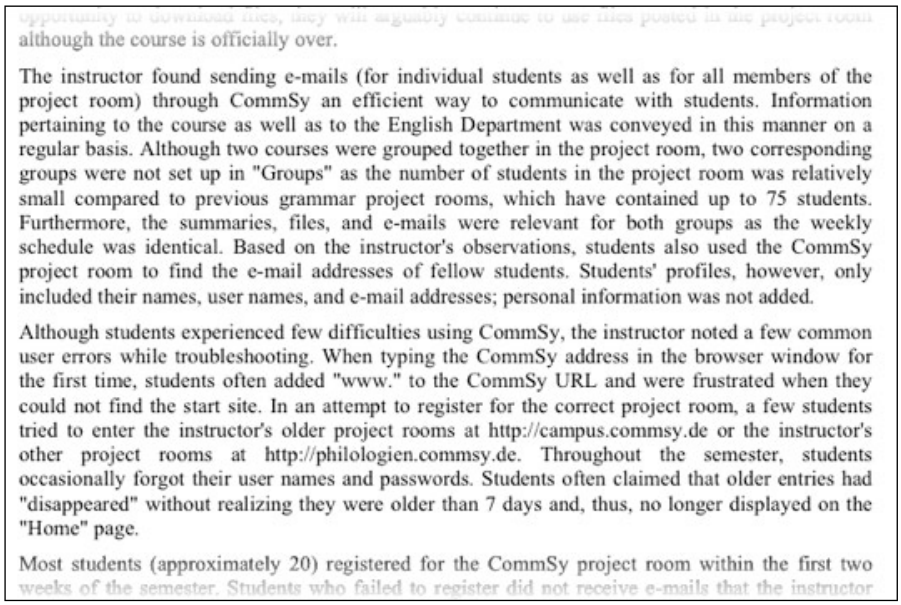
While their format differs, the former being a variation of traditional PD workshops and the latter a form of documentation for both users and developers, both techniques share a shift in the level of discussion: as a shared language had to be found for users from different contexts, the analysis of individual contexts and tasks[75] would not be accessible to users from different communities of practice. Instead, the exchange with members of other communities of practice encouraged users to reflect their own usage; the confrontation with an external perspective made it possible to reflect aspects of use that were unquestioned within their community of practice: „Challenged by other participants to explain why, e.g., certain features are important to them, and contrasting this with experiences from other backgrounds, they start to think through and sometimes question their use routines." (Obendorf, Finck, Janneck, & Bødker, in prep.)

Intercontextual user workshops are very similar to existing PD workshops: the development team invites interested users to participate in design sessions. An important difference is the background of the invited user representatives: while PD tries to include different stakeholders that share a single application domain, intercontextual user workshops explicitly communicate that many different users with diverse backgrounds will attends. These users cannot easily exchange information about their work processes—and they are also usually not very interested in learning how other professions work, at least when the usual level of detail necessary for a task analysis is reached. Instead, they attend the workshops because they would like to know how to make better use of the software, and help to make the software better. As the latter is usually linked to a better fit to their individual use context, intercontextual user workshops must find a way to resolve the conflict of interest as different users try to ‚push' their context. The solution found in CommSy development is a focus on development values, which are formulated so universally that they can be applied to many different contexts.

The discussion in intercontextual user workshops thus focuses on the negotiation of values. At the same time, concrete features are the object of discussion. The argumentation of users changes accordingly: if they would before have argued „we need feature X to do task Y", they now try to express their needs in more abstract terms, „feature X is necessary as it is in line with value Z". Simplicity plays an important role in these discussions as users base their requests often on this value—and after the connection with simplicity has been made, short excursions into individual use contexts are the rule to explain why feature X makes the software simpler. After realizing the connection with a shared value,

75. Workflows—often institutionalized in large organizations—are the preferred object of analysis for PD techniques. As different contexts usually don't even have common tasks, their analysis becomes less useful. As an additional factor in the case study, virtual organizations commonly lack well-established structures and procedures, and are consequently much harder to analyze (Janneck, & Finck, 2006; Janneck, Finck, & Obendorf, 2006a).

however, other users tend to be much more sympathetic of the difficulties of users from other domains. Even more so, they usually find value in the intelligent misuse of the software by other users if they first realize the value created for these users.



although the course is officially over.

The instructor found sending e-mails (for individual students as well as for all members of the project room) through CommSy an efficient way to communicate with students. Information pertaining to the course as well as to the English Department was conveyed in this manner on a regular basis. Although two courses were grouped together in the project room, two corresponding groups were not set up in "Groups" as the number of students in the project room was relatively small compared to previous grammar project rooms, which have contained up to 75 students. Furthermore, the summaries, files, and e-mails were relevant for both groups as the weekly schedule was identical. Based on the instructor's observations, students also used the CommSy project room to find the e-mail addresses of fellow students. Students' profiles, however, only included their names, user names, and e-mail addresses; personal information was not added.

Although students experienced few difficulties using CommSy, the instructor noted a few common user errors while troubleshooting. When typing the CommSy address in the browser window for the first time, students often added "www." to the CommSy URL and were frustrated when they could not find the start site. In an attempt to register for the correct project room, a few students tried to enter the instructor's older project rooms at http://campus.commsy.de or the instructor's other project rooms at http://philologien.commsy.de. Throughout the semester, students occasionally forgot their user names and passwords. Students often claimed that older entries had "disappeared" without realizing they were older than 7 days and, thus, no longer displayed on the "Home" page.

Most students (approximately 20) registered for the CommSy project room within the first two

Figure 103: Example of a case description in the commented case studies. (Finck et al. 2006)

In contrast to the direct and immediate interaction possible in intercontextual user workshops, the commented case studies require more effort in both creation and reception (Finck, Janneck, Obendorf, & Gumm, 2006). In return, they deliver a more complete, and more persistent vision of use and development (compare Figure 103). They consist of user-contributed descriptions of actual use, complete with suggestions for future development and ‚lessons learned' for using the system, an access key to support quick reading, a system description, and clarifications and comments by the development team (Finck, Obendorf, & Pape, 2004; Finck & Janneck, 2006). For developers, they bundle authentic, unedited reports from different use contexts and help them develop a more thorough understanding of their requirements (Table 21). Users are given access to the experiences of other users—and to the developers' motivations for design decisions. It thus becomes possible to anticipate future design decisions.

Table 21: Aspects covered in the different sections of the commented case studies. (Finck et al. 2004)

Documentation appeals to themes of	access key	description	system	example cases	reflection	design
Functionality			●	●		●
Use / Procedure	●			●		
Goal / Value	●		○	○		●

Both techniques were implemented with some success, and although users found it initially difficult to discuss their work practice on a more abstract level, they were very interested in exchange with practitioners from other fields; a direct exchange of reflective considerations about their work practice became possible. Reflection on system usage and design happens both on a concrete level of use practices and tasks within specific communities and an abstract level of design philosophy and underlying viewpoints and values within the different contexts. By means of addressing values on an abstract level, users from different contexts succeed in clarifying their requirements without having to share details of their daily work routines and practices. An example: The demand for highly differentiated access rights, which was expressed by several users, boiled down to a discussion of trust, hierarchies, and authority and the value of equality that is inherent in the design of CommSy.

It must be borne in mind, however, that even dedicated value negotiations do not guarantee the success of cooperation; in the case of Virkon, the participative development with one virtual network frequently tripped over an apparent mismatch of proposed values, and concrete requests for functionality (Janneck, Finck, & Obendorf, 2006b); it turned out that the network's self-conception—that was a good match with CommSy's design principles—did not relate to the actual work practice within the network.

6.3.3 Discussion

In the discourse of practitioners, a dichotomy is often present between „design" and „construction" (Berkun, 2005), or „creative design" and „engineering design" (Löwgren, 1995), with the former referring to an explorative approach of solution-finding, and the latter of implementing according to specifications. Perspectives based on this dichotomy are restricted as they do not realize that design requires construction, and construction requires context. Design has logical elements as much as construction is often a creative activity (Gedenryd, 1998). The development of the CommSy software is an example for a process where design and construction are intertwined to the point of being inseparable—development is considered an activity that can be perform by non-programmers. An unconventional aspect of CommSy's development process is that users are readily accepted as members of the development team, and many developers are frequent users, as well. Together, these conditions allowed a transformation of the ‚we-need' feature discussion to a reflective ‚what-is-the-tool' discussion, and to the formation of *design principles* early on in the development process. The shared understanding of what the tool incorporates, borne by the tradition of informal corridor communication, and supported by regular workshops, focused development on the question of how the tool could best be used in existing, and later in new, contexts.

During the early and the golden years, CommSy development was mainly focused on a single context. The meaning of ‚simplicity' in these years encompassed both *functional* and *structural minimalism*. This led often to open conflicts where advocates and opponents of

a new feature both based their argument on ‚simplicity'. The notions of minimalism put forward here allow a disambiguation of simplicity in many of these conflicts.

When CommSy development lost its single focus and targeted new contexts, and thus different communities of practice, *functional minimalism* became more and more difficult to follow as a value. Simplicity remained a core value, although initial design decisions based on *functional* simplicity were revised. In return for this partial loss of meaning, simplicity acquired new interpretations; the design principle *media mix* was strengthened and renamed to *openness*. The simple use of CommSy as ‚a tool interfacing with other tools' moved into the focus of development, allowing the functional core to remain relatively stable, but requiring structural extensions and external interfaces.

Due to the coverage of several distinct use contexts, the aspect of *compositional minimalism* was also strengthened in the development process—CommSy appropriation had to be supported to allow a continuing development of a single software package, maintaining a relative simplicity in functionality, and still cater for a wide variety of different uses. A *community of interest*, enabling users from different contexts to communicate with each other and the development team, had to be established to allow further participation of users in the development process. *Value-based participation* used the existing principles to shift the design cooperation to a new level that allowed the comparison and categorization of practices from different contexts.

Based on the different levels of abstraction that we find in both the workshops and the case studies (Fig. 102), we distinguish three levels where our software development process benefited from the methods described here (see Table 22): on the first level, users' acceptance of development decisions is increased as they learn about the rationale behind design. Reciprocally, developers obtain legitimation as their underlying values are being confirmed by users. On the second level, empowerment of users comes along with a deepened understanding of the respective domain on the developers' side. Building upon this, on the third level, users are enabled to integrate the software into their work practice, while developers can form a consistent vision spanning the different use contexts.

Table 22: Benefits of value-based participation. (adopted from Finck et al. 2006)

Users	Development Team
Increased Acceptance	Legitimation for design
Empowerment	Domain understanding
Use vision	Software vision

Rising the communication to a more abstract level risks introducing new sources of misunderstandings. It is thus crucial for the communicating stakeholders to negotiate the meaning of the values that communication is based upon. A differentiated understanding of minimalism reduces the risk of misunderstanding simplicity—as the previous discussion demonstrated, the number of different interpretations is high. Minimalism is an im-

portant tool to raise the awareness of what simplicity means, and when a shift in its meaning takes place. This refined understanding of simplicity can help to implement the proposed technique of value-based design successfully.

For many applications, it makes sense to assume that the software development process is complemented by a software appropriation process on the users' behalves. If one accepts that the designers' power to determine the software does not extend into the users' application domain, it becomes an important aspect of development processes to learn from the users how they actually use the functionality that was intended for a specific purpose.

6.4 Reflection

This chapter lists several approaches to create better development processes using the notions of minimalism developed in this thesis. As the analysis of existing designs in the last chapter demonstrated some value in using minimalism as an analytic perspective, the first approach tries to formalize and make more accessible the minimalist standpoint in form of a design game. This was successful for discerning *functional* and *structural minimalism*, but the concepts of *architectural* and *compositional minimalism* were not immediately understood by the students who served as designers in the informal study. As the game is restricted to use design prototypes and other material available in form of paper, these notions were difficult to integrate even by more experienced designers as they are by definition determined by design context external to the system. It thus seems to be more feasible to restrict the use of the game to differentiate between structure and functionality, and find other means to address the contextual forms of minimalism.

In search of methods that would focus on the latter forms of minimalism, the CommSy development process is analyzed as a case study. Due to the dynamics of the development process, a shift from functional and structural to architectural and compositional minimalism was induced. Although this resulted in a somewhat increased complexity of the product, in some areas *architectural minimalism*, exemplified by placing functionality into optional modules, and a turn towards openness to other tools, also reduced the overall complexity of the system. As new contexts were constantly being added to the user base, it became clear that not all use situations could be planned for. Instead, support for appropriation became central, both for individual communities of practice, and for a community of interest centered around the product. To this end, value-based development evolved as a methodology that enabled community building and participatory design for different fields of use—*minimizing composition* within the design.

To allow a generalization towards other development processes, the focus was consequently laid upon scenarios as a popular design technique. Apart from informally representing context, thus grounding the design, scenarios tend to reduce the design in several aspects: due to their incomplete representation of context, good scenarios tend to capture prototypical interaction—allowing for an optimal structural fit. Their restricted form limits the number of features described within a scenario, thus identifying core functionality. If used to guide development, this can help avoid integrating too many features, al-

though feature creep may still apply. Scenarios thus act as a *filter for design*, reducing *functionality* and *structure*. If applied to this end, they can also support the partitioning of functionality and be used to identify the building blocks for an *architecturally minimal* system.

Finally, scenarios are put to the test in a new form of development process that is a blend of extreme programming and usability engineering. Although a number of usability techniques are integrated in the process, scenarios fulfill the function of distilling the requirements gathered into a minimal form, and generate a design solution that both solves the formulated problem and is extensible to include more conventional features. This challenge is successfully met by the adopted scenarios.

The minimalist standpoint can be considered a versatile tool for analyzing and modifying development processes—the nature of applying the notions of minimalism displays a large variation in the presented approaches. Although the empirical basis is far less broad than in the last two chapters—understanding development processes as blueprints, not products, limited discussion to those techniques where personal experience was available—the exemplary use of minimalism to identify and influence dynamics in development processes demonstrated the value of a constructive use of minimalism.

Limiting Factors for the Selection of Case Studies

Both software and usability development processes are often taken as finished products. They are published as standards and in books or manuals, suggesting that they provide well-defined procedures. However, experience indicates that each development process is different, and the ‚processes' defined based on experience and research should rather be understood as process blueprints whose actual realization unfolds as the process is implemented (Floyd et al. in Pape, Krause, & Oberquelle, 2004). While the last chapter intended to illustrate the breadth of implied meanings of the individual notions of minimalism by analyzing manifold designs, this has limited the discussion in this chapter to those methods that have been personally experienced and evaluated by the author.

7 Reflections on Minimalism

Bad artists copy. Good artists steal.
— attributed to Pablo Picasso

The previous chapters established a minimalist standpoint for the design of interactive systems: an analytic perspective of existing values and processes in human-computer interaction based on four notions of minimalism, combined with a preference of the simple. It is now time to question what benefits and limitations the preceding discussion has managed to identify. Has something new been generated by the transfer of the notions of minimalism into the realm of interaction design? And has something else been lost, or become less visible? The following paragraphs try to give an account of how the notions of minimalism have influenced the perspective of analysis.

7.1 The Minimal Perspective on Design

A beautiful program's way of doing things is so close to your own that creative symbiosis develops, a thought-amplifying feedback loop.
— David Gelernter (1998, 25)

The complexity gap, the chasm between the desired ease of use and the necessary complexity of use environments, or, more precisely, the reduction of the gap, is an aim of numerous methodological approaches to the design of human-computer interaction. Yet, while the design practice is busy trying to build bridges spanning the complexity gap, a reflective treatment of the values and motivations that underlie these approaches is rare. Instead, a shared belief that simplicity is desirable forms the basis for manifold design decisions and suggestions. The meaning of simplicity, however, differs from author to author, and even from application to application, and the phrase ‚less is more' is used wherever it serves an illustrating function.

Minimalism, closely connected with the ‚less is more' motto, has not been defined as a term within human-computer interaction before. Its definition bears the promise of a deeper understanding of simplicity, and of a differentiation between different notions of simplicity that overlay and obstruct each other in the literature. Before a successful defini-

tion, however, stands the transfer from the liberal arts, spiritual home of the term, to the interdisciplinary field concerned with the design of interactive systems. A proposal for this transdisciplinary undertaking was made in this thesis, introducing a minimalist perspective to the science of design.

The roots of minimalism lie within painting and sculpture. Conceived of by critics, minimalism was used to identify a rising group of artists in the U.S. during the 1960s that concentrated on experimenting with different means of reduction (2.1). They tried to reduce their *means*, the *meaning* of their works, the structure in their works, and the constructional effort by using *patterns*; different approaches alternatively tried to minimize actions of the spectator, delivering total control, and creating *freedom* through involvement, turning the onlooker's interpretation into an important part of the artwork. The term was so successful that its use spread to music (2.2), where different artists created modern „classical" music, opposing serialism, and creating a new musical language that reintroduced tonality, and employed different forms of repetition. Although minimal music itself never became part of popular culture, its influence extends to current pop and electronic music.

Minimalism was also taken up to describe trends in literature (2.3.1), typography (2.3.2), and architecture (2.3.3). While each discipline had its individual interpretation of the term, and fierce battles were fought defining the artistic merit of reduction, almost all protagonists agreed on the listed aspects. The subsequent transfer of the notion of minimalism to the design of interactive systems is thus based on a shared, although not univocal, understanding of the term (2.4).

Minimalism is both a useful and a dangerous term for design as it has found its way into everyday language. Although it easily engages members of diverse background into discussion, negotiations of meaning precede almost every discussion of minimalism as incompatible meanings must be excluded. The definition of minimalism chosen in this thesis is thus backed by a definition of the conceptional background of this work (3.1), and preceded by a number of possible interpretations that were found to be unhelpful for analyzing designs (3.2). Four notions of minimalism are defined (3.3.1–3.3.4), and their interactions are discussed (3.3.5). All four notions of minimalism refer to the interface seen from a user's perspective; although the definition uses the terms structural and architectural, which are usually connected with the internal construction of a system, the focus of this thesis lies on the interaction with a design, examining the construction of a system only as it becomes determinant of the perceivable interface. Minimalism is explicitly more than superficial, yet it deduces requirements for the system architecture from the architecture of the interface, not otherwise.

> *Functional Minimalism* denotes the reduction in accessible functionality. A functionally minimal system focuses on its core competency and presents only few key features to the user.

Structural Minimalism denotes the reduction in perceived access structure. This involves trade-offs as important functionality is made more directly accessible—at the cost of making unimportant functionality harder to access. The success of this strategy depends on the ability to determine this importance; adaptive approaches must consider the negative effects of changing access structures.

Architectural Minimalism denotes the reduction of perceived complexity through a transparent distribution of responsibility across tools. The combination of simple tools creates the necessary functional complexity for the user.

Compositional Minimalism denotes the reduction of a design's specificity for planned tasks. A compositionally minimal system makes fewer assumptions about its use and places less restrictions upon its users. This extends the notion of user control beyond the individual task.

Although there is some conceptual overlap, the different notions of minimalism focus on different aspects of the design: the former two highlight aspects of the concrete design, while the latter two point towards more transient aspects of the design that are determined by the construction method and the introduction into the work context. Functional minimalism and structural minimalism both affect the system's *user interface* directly, while architectural and compositional minimalism work at different levels, namely the inner construction of the system and the users' ‚interface' to the system, or the integration with other tools. Structural and architectural minimalism both focus on adapting the tool to the task, modifying the structure by which functionality is accessed, while functional and compositional minimalism stress the use of the tool and the context of use.

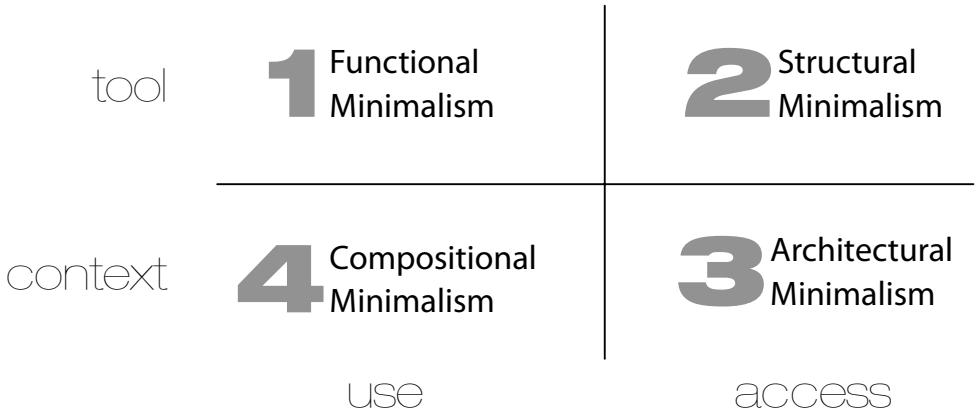


Figure 104: The four notions of minimalism and their respective design focus

Together, these four notions of minimalism form a perspective on existing, experience-based expert knowledge that allows to group and categorize design heuristics that have been found to support the design of simple interactive systems. By differentiating between qualities of an interface that in combination create a simpler use experience, a more constructive understanding of interaction is defined. Compared with existing standards and norms (4.2), a new perspective on existing HCI lore is created that groups and categorizes often cited guidelines without aspiring to provide a complete theoretical frame-

work (4.3–4.4). The minimalist perspective is linked with an understanding of design as a craft (4.5), and provides an alternative to the notion of consistency more adequate for those designs where becoming an expert is not associated with coping with increasingly arcane features (4.6). The four notions of minimalism that were defined based on parallels with art and music (3.3) thus demonstrate their theoretical value and are refined as existing HCI lore revealed missing details (4.7).

7.2 Minimalism as an Analytic Tool

> Things should be as simple as possible. But no simpler.
> — attributed to Albert Einstein

The analysis of design products receives intense attention in this thesis (5.1–5.4): the four notions of minimalism are each introduced with an ‚exotic' example that highlights peculiarities of the respective notion before real-world examples are introduced and discussed. In total, twelve examples are examined for their minimal qualities, with different application for word processing providing the background for an encounter of the different perspectives on a single domain.

Each product was chosen for its simplicity, and the four notions of minimalism are used to discuss how this simplicity relates to the different perspectives on reduction. Although each example is simple in more than one way, the four perspectives allow a differentiation of the minimal aspects that combine to create the design's simplicity. It is also possible to speculate about design priorities that led to the development of the individual product.

Among others, the CommSy platform, also basis for the case study identifying process genres (6.3), is introduced and demonstrates the impact of a waning focus on functional minimalism. An explanation model for the initial success and the current stagnation of the Palm Pilot (5.2.2) is put forward, and a historical episode of Nokia building explicitly simple mobile phones (5.2.3) illustrates the difficulties of keeping pace with the feature frenzy. Some suggestions for causes of Apple's current good fortune (5.1.2, 5.3.2+4) are made, and parallels are drawn to current trends in the World Wide Web (5.3.5, 5.4.5). Different approaches to standard office software for word processing (5.1.4, 5.2.5, 5.3.6, 5.4.6), demonstrate the consequences of pursuing individual minimal qualities for design. The necessity of including the appropriation of products by users as an parallel process during development is highlighted as successful product are explored that allow its users to freely choose the use context, e.g. presentation software (5.4.4) and WikiWebs (5.4.5).

The discussion demonstrates that it is possible—and that it can be very profitable—to build minimal interfaces—even if customers demand tools for complex and diverse tasks. The examples focus on specific qualities of existing designs and relate these to the notions of minimalism, identifying minimal traits in the examined designs.

But the examples serve not only an illustrating function, they are also used to question and improve the abstract definitions of the notions of minimalism. As a consequence, the minimal terminology is further sharpened, allowing a shift towards the constructive to be

introduced that enables not only the abstract designation of minimal attitudes, but also the identification of matching minimal ‚best practices'. These design heuristics are extracted and assigned to one of the four notions of minimalism, in the hope that they could be helpful for new designs (5.5.3).

An abbreviated overview of the minimal design heuristics given below in Table 23. The listed heuristics are not understood to be imperative. Rather, it is obvious that heuristics assigned to a single notion of minimalism contradict each other, e.g. suggest that a structural reduction could be reached by prioritize access according to frequency, or activity. This is no fault—instead, the heuristics describe alternative approaches to attaining an effect that is covered by the respective definition of structural minimalism. A designer must therefore take the design context of the given examples into account, and individually weigh trade-offs for his specific design problem.

Table 23: Summary of Minimal Design Heuristics (compare 5.5.3, Table 19)

Functional Minimalism
Identify the ***core functionality*** of the design. Define whether the design aims to be a simple tool.
Take a holistic perspective: define the responsibilities of the design ***as a whole***.
Discuss trade-offs involved with extending functionality. Keep track of the initial vision and its changes.
Assess the feasibility of providing multiple interfaces using a single platform (optionally: let users select).
Structural Minimalism
Prioritize access according to ***frequency***.
Prioritize access according to ***activity.***
Translate prioritization into appropriate modalities.
Architectural Minimalism
Examine your interface architecture at different levels: Building blocks can be gestures or interaction techniques, or manifest themselves as visible tools.
Modularization, or division in tools risks fragmentation—the ***whole design must remain visible***.
Compositional Minimalism
Designs should not be generic, but generalizable. Do not try to plan the future, support „misuse".
Functionally minimal designs are one approach towards design support for appropriation.
Modularize functionality in your design—form „building blocks". The resulting „use patterns" can be „interpreted" in different use situations by users.

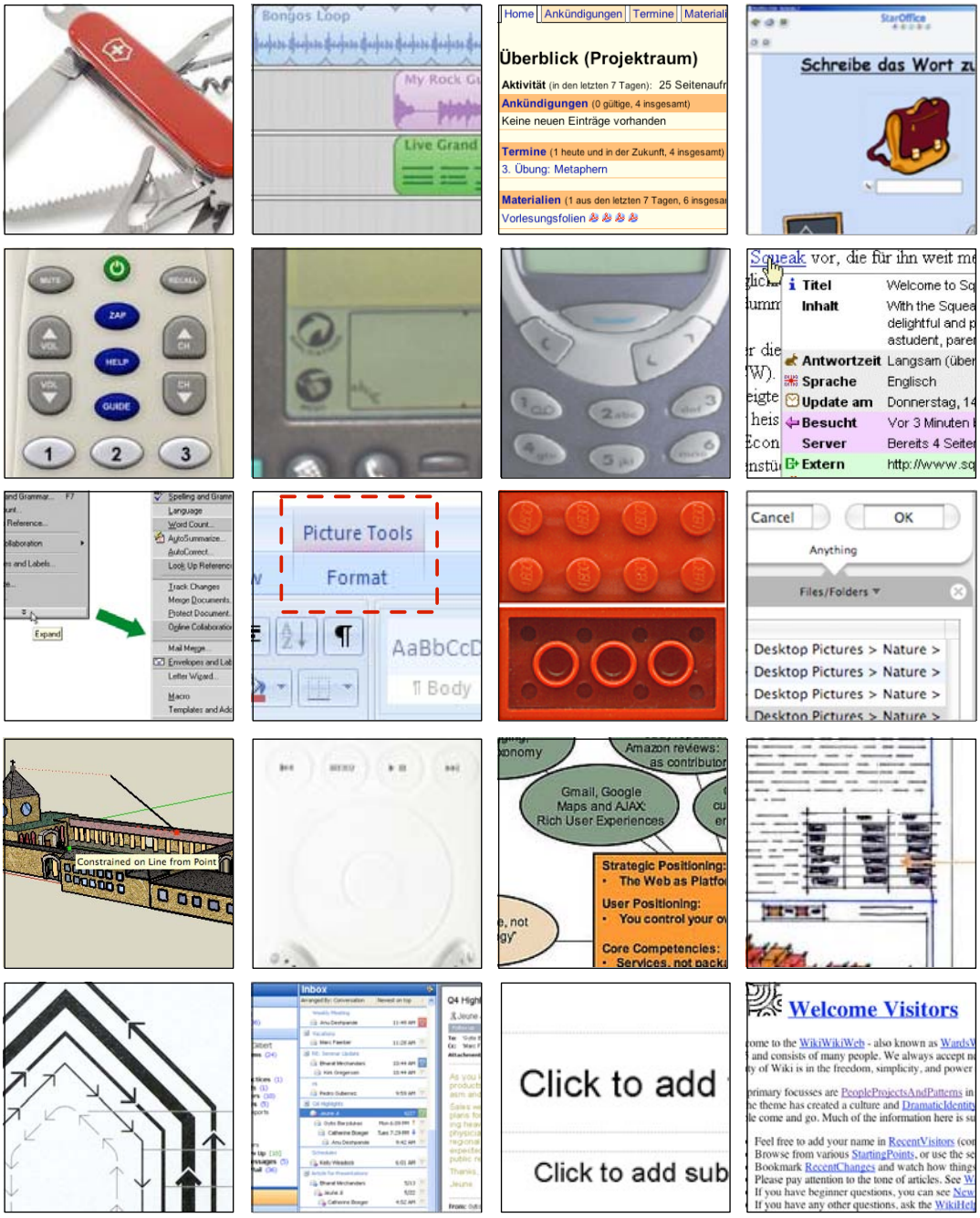


Figure 105: A visual index to the examples contributing to the minimal analysis

Figure 105 displays an overview of the products examined in the fifth chapter. *First row*: 5.1.1 Cutting edges, 5.1.2 Apple GarageBand, 5.1.3 CommSy, 5.1.4 functionally minimal word processors. *Second row*: 5.2.1 Remote controls, 5.2.2 Palm Handheld, 5.2.3 Nokia 3110 series, 5.2.4 Hyperscout. *Third row*: 5.2.5 structural minimalism in Word 2000 and 2007, 5.3.1 Lego bricks, 5.3.2 Apple Automator. *Fourth row*: 5.3.3 SketchUp, 5.3.4 Apple iPod, 5.3.5 Web 2.0, 5.3.6 architectural minimalism in Ragtime. *Fifth row*: 5.4.1 Learning Buildings, 5.4.3 Email, 5.4.4 Powerpoint, 5.4.5 WikiWikiWebs.

7.3 Minimalism as a Constructive Tool

> Perfection is reached, not when there is no longer anything to add,
> but when there is no longer anything to take away
> — Antoine de Saint-Exupéry (1900–1944)

The successful application of the four different notions of minimalism in the analysis of design products created an immediate motivation to give them a more constructive role in the designs of interactive systems. As they could be used to differentiate and prioritize aspects of ‚simplicity' in designs, an immediate idea was to use the same language in system construction like in system analysis; the resulting technique was the Minimal Design Game (6.1). To enable a more subtle integration in existing development processes, existing techniques can be analyzed and selected with regard to their minimal qualities; this allows designers to think about the design instead of having to concentrate on methodology. The XPnUE process that integrates techniques from usability engineering and extreme programming is presented as an example for this approach (6.2). Finally, the focus on architectural, and specifically on compositional minimalism is served by the process genre of *value-based design* (6.3).

7.3.1 The Minimal Design Game

The Minimal Design Game (6.1.1) is a paper-based set of methods that tries to first analyze an existing design for possibilities for reduction, and in a second step revisits the reduced design ideas and assesses their suitability for integration with the core idea that was uncovered by stripping away the unnecessary.

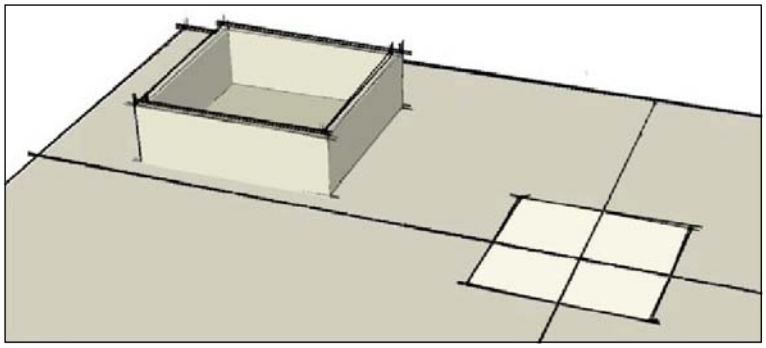


Figure 106: The Minimal Design Game Setup: Ready for Reduction?

In a preliminary evaluation of the Minimal Design Game (6.1.2), most participants immediately understood the notions of *functional* and *structural minimalism*, and produced promising results using the provided guiding questions. Interpreting *architectural* and *compositional minimalism*, however, was more difficult; these notions are not obvious and present a new perspective, they both make use of terms that are already in use in computer science, and specifically in software development. As a second consequence from the evaluation, it became clear that the demonstrative focus on minimal designs encourages critical assessment about how minimal a solution should be.

7.3.2 XPnUE: Fusing Extreme Programming and Usability Engineering

Most designers like to focus on their design without constantly assessing which theoretical stance they assume; design techniques are often chosen for their known qualities, and according to their match with intended product qualities. An assessment of the reductionist tendencies of scenario techniques in minimal terminology (6.2.1) provides an elegant method to pursue minimal designs: an existing development process can be changed using specific types of scenarios if e.g. a functionally minimal design is desired.

Although agile development processes explicitly aim at reduction, their focus does not overlap with the notions of minimalism defined in this thesis. Instead, reduction in agile processes focuses on the construction of a system, and the construction process. By avoiding the development of unnecessary components, incrementally advancing towards an ideal solution, and minimizing the necessity for process documentation a lightweight process is created that demonstrates its agility in reacting to requirement changes, and thus supports the development of software where a formal specification is not possible in advance—most interactive systems would fall under this category. However, agile methods, such as Feature-Driven-Design, strongly focus on individual features to negotiate and prioritize product design. While this approach promises to minimize risks for both developer and customer, the lack of a holistic focus as maintained by a minimalist standpoint risks the fragmentation and over-diversification of the design.

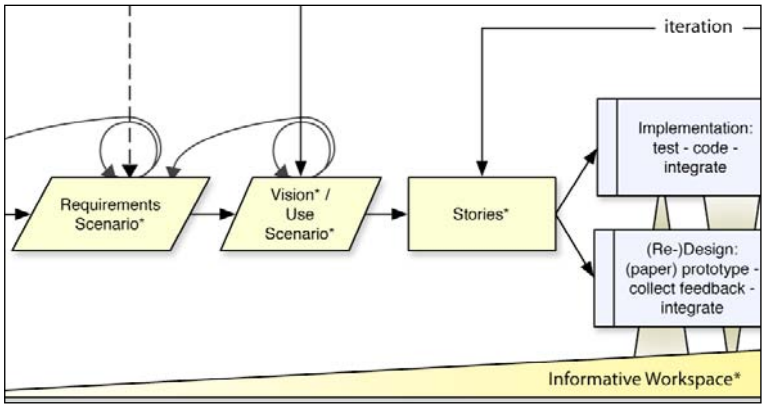


Figure 107: The XPnUE process pattern fuses Scenario techniques with Extreme Programming (6.2.3)

XPnUE is a process model that was developed to fuse Extreme Programming (XP) with Usability Engineering, or more specifically, with scenario techniques that focus the design on a core task, and to provide a unique and novel solution not only in technical terms, but also on the level of interaction. To this end, scenario techniques are used early in the process to focus the analysis of existing practice on a core set of tasks, and extract requirements for supporting key problems in scenario form. Creative solutions for the identified core functionality are then generated, and a decision about the direction of development is made and documented in form of a vision. A context-based scenario of future use is inserted in the process before the XP story technique is used to negotiate fine-grained iterative implementation tasks. To keep the focus on the interface, the stories are enhanced with storyboards, graphical representations of system and use. The informative work-

space assumes an even more central position than in the XP technique, collecting not only information about the process state, but also about contextual investigation, elements of the design vision, and design rationale.

Experience from implementing the XPnUE process pattern in a post-graduate course indicates that despite technical and temporal difficulties, the selected scenario techniques were very successful in capturing and defining the core functionality for the design activity (6.2.3). Specifically, the holistic perspective introduced by the focus on key functionality helped to keep development in track, and allowed our students to deliver an innovative solution in form of a working prototype.

7.3.3 Value-Based Development

Finally, the identification of shifts from focus on functional and structural to architectural and compositional minimalism in an example development process demonstrated that the minimalist standpoint is a useful tool also for identifying process genres. The development process of the CommSy community system was examined as a case study (6.3.1). Created in 1999, CommSy is today developed as an open source project after many organizational changes, and a continuing broadening of use contexts. These new use contexts induced a change in the development focus: while early on, decisions were made to keep the software simple in features, and it was possible to create an access structure optimally adapted to the single use context, the new users demanded new functionality, and used the software in different ways.

The reaction of the CommSy developers was an implicit shift in the values that guided development: in minimalist terminology, the ‚simplicity' that was defined early on in the development process changed its meaning from functional and structural to architectural and compositional minimalism. This design shift was associated with changes in the use of development methods: immediate feedback was more difficult to gather from the diverse communities of use, and using traditional techniques for participatory design became more difficult due to the influx of new users from diverse backgrounds. As a consequence, the design discourse shifted to a more abstract level and became based on values.

Value-based development aims to negotiate shared values among the users and developers (to take into account the artifact's design history). This implies that participatory design evolves no longer from the procedures that determine a user's practice, but focusses on the values and goals that users pursue as these can be shared between different use contexts. The abstract values can be related to individual features of the software system, and the discussion is thus located on both a very abstract, and a very concrete level. By involving some user representatives directly in *intercontextual development workshops*, and in developing the *commented case studies*, a multiplication effect is created, and many other users can indirectly participate in the development efforts (Figure 108). This shift in methodology makes it possible to target a system that is no longer structurally minimal—as there is no single use case that it could be optimally fit to, but instead to identify blocks of functionality that can be made into tools for specific audiences—creating a more archi-

tecturally minimal interface, and to examine where CommSy inhibits ‚creative misuse', thus reducing unnecessarily strict composition in the design.

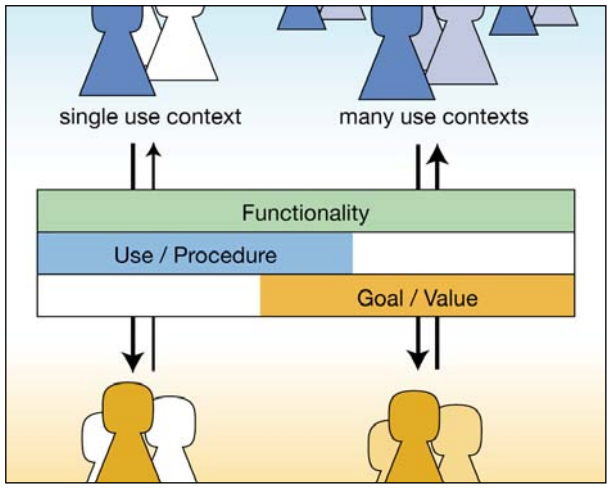


Figure 108: Value-based development accommodates diverging needs of manifold contexts.

Minimalism has a twofold role in the examined use case: it is used to identify the shifts in the development process, but it is also a tool to clarify values inherent in the design, thus improving the communication with users. While the notion of simplicity suffered from being ill-defined, and many situations occurred where contradicting arguments were both based on the simplicity value, the four different perspectives created by minimalism allow to explain the conflict, and replace the clash of interpretations with an informed choice.

7.4 Unconnected Ends

Some limitations of the approach taken by this thesis must be brought into the light before drawing conclusions; this is combined here with an agenda for further research.

Mathematicians dislike computer-assisted proofs—correct and verifiable as they may be, they don't transport the spark of intuition that seems to be required from scientific mathematical practice. Computer scientists like concepts-of-proof to demonstrate scientific progress, while art historians claim that science must take the form of well-documented arguments. While transdisciplinary work is often called for (e.g. Mutius, 2004), the different understandings of how science works can make it difficult to transfer concepts (comp. Welzer, 2006). In this thesis, the transdisciplinary approach, aiming at the transfer of the notion of minimalism from art and music history to the field of human-computer interaction, made necessary some unusual ‚design decisions': there is no single interpretation for minimalism, there is no measurement of degree, and there is no constructed (read artificial) prototype system demonstrating the meaning of minimalism. Instead, this thesis has tried to preserve the inherently discursive character of the notion of minimalism in the transfer to prevent a loss of meaning.

An inherent limitation lies in the notion of minimalism, that in the arts never acquired a canonical state. The transfer of the notion of minimalism can thus only be expected to

start a discourse about its meaning: the notion of minimalism must be open to new interpretations in HCI. The four notions proposed here were chosen as they form a solid foundation for the analysis of designs, and can to some degree be clearly differentiated. They should by no means be understood as the only possible interpretations of minimalism, or as the only meanings of simplicity.

The empirical base forming the discussion of designs and processes is necessarily limited; specifically, more detailed analyses of designs—this thesis chose breadth over depth here as internals of designs are not easily accessible[76], and more development techniques and processes created based on an informed choice of minimal design qualities are necessary to prove the practical value of minimalism. The Minimal Design Game needs to be evaluated in real-world conditions; a second version design would be most interesting as it forces the focus away from the rather trivial notion of functional minimalism (5.1.5).

For *human-computer interaction* as a discipline, effects of minimalism for both the development of standard software—with open-source development presenting a special challenge (6.3), and the increasingly important field of configuration and end-user development (5.3, 5.4) require further research. The process of appropriation needs further integration in the reflection of usability methods (5.4, 6.3). To encourage the transfer to *design practice*, more examples, and more experiments are necessary, transforming the minimalist standpoint in a set of practical tools.

Software engineering must find answers for its relation to usability engineering, and face the focus on the whole that is promoted by architectural and compositional minimalism. The analysis in this thesis indicated that although both disciplines aim to improve software quality by attaining simplicity, the focus of reduction differs. The use of techniques from usability engineering in software development processes promises to allow an increased focus on the whole design, thus increasing the reflection on the necessity of features, and the forces that drive and orient development. An equitable fusion of techniques from these two disciplines might create synergies for the design of simple systems.

Finally, to demonstrate the attractiveness of these techniques for professional software development, the economical use of minimalism must be discussed, and a possible role be defined for minimalism as part of internally or publicly communicated company values (Jenson, 2002; Jenson, 2005). Research on the function of simplicity in marketing needs to be integrated—a starting point lies in economists' discussions the role of simplicity in management and business processes (Jensen, 2000, 31; Trout, & Rivkin, 1998), and for marketing[77] (Cristol, & Sealey, 2000).

---

76. Breaking with scholarly tradition, sources from the Web have been cited where no ‚scientific' source was available—as designers usually do not reflect openly on their practice, Weblogs and press releases were used.

77. Minimal marketing must market the minimal; instead, it has been misunderstood as investing minimal effort in marketing, assuming that „demand will grow … simply because they are offering it" (Kotler, 1975, 8), placing „confidence in the assumption that quality speaks for itself" (Bittel, & Bittel, 1978, 709), and, e.g. by airlines, as „focusing simply on the bare product, the price and easy booking" (Stone, & Foss, 2001, 355).

7.5 Conclusion

There is always an easy solution to every
human problem – neat, plausible and wrong.[78]
— Henry Louis Mencken (1917)

Beauty is Simplicity plus Power. Simplicity is desirable. And simplicity is created through reduction. Based on this understanding of design, answering the questions regarding direction and degree of reduction, and thus differentiating the meaning of simplicity was motivation for the transfer of the notion of minimalism to the design of interactive systems that this thesis attempts to implement.

This thesis does not attempt to provide an unambiguous set of rules for design. Instead, it does something more: The minimal standpoint that was introduced in this thesis offers a new perspective on the design of interactive systems: focus shifts from individual features to the overall Gestalt of the design. This emphasis on the whole promotes a holistic perspective: design evolves no longer solely around the *user*, but converges on *use*. The consequential perspective of *use-centered design* presents an alternative to focusing alternatively only on the user, or only on the technical composition. The focus on reduction, and the simultaneous emphasis on the whole promises to keep the expansion of complexities within reasonable dimensions, and to human scale.

Minimalism adds constructive advice and support in choosing concrete procedures to this abstract posture. Replacing minimalism with simplicity, four values are defined in this thesis that can guide the design of interactive systems: *functional* simplicity, *structural* simplicity, *architectural* simplicity, and *compositional* simplicity. These values differentiate the meaning of simplicity, and can thus be used to analyze existing designs, or make informed design choices when developing a new product. The minimal perspective was also shown to be useful for analyzing and modifying the processes that are used to install product qualities in designs, both on a literal, and on a very abstract level.

It is my hope that minimalism has the potential of widening the reader's perspective on the design of interactive systems, and that the transfer of a notion rooted in art and music was successful insofar as it has brought a new standpoint to design that can help both scientists and practitioners to better understand their own discipline, and reflect on their design practice.

Perhaps, it is more than coincidence that Microsoft as the manufacturer of some of the most complex applications promotes „powerful and simple" as one of five design principles for its new operating system *Windows Vista* (Microsoft, 2006a), and that Apple is so successful selling its complex systems with an air of simplicity. It might indicate that simplicity, and thus minimalism, is finally recognized as an important basis for ease of use, and perhaps, we are going to see „more" of „less" in the future.

---

78. On the Web, you will often find the more suggestive, albeit incorrect quote „For every complex problem, there is an answer that is clear, simple—and wrong".

8 Appendix: CommSy Interviews

The CommSy system is used throughout this thesis as a case study. As the author never was a member of the development team, evaluation is based on direct and indirect accounts of the primary developers. In order to supplement the information about the CommSy project available through a number of sources, including dissertations, books, conference articles, and personal communication with specific data about the use of simplicity as a value, a number of interviews were held from April through August 2006.

As a high degree of openness in interviews was desired, but a number of external factors required a high efficiency of the interview process—the time of most interviewees was strictly limited—a compromise in regarding the structure of the interview was necessary. The resulting format of interviews was semi-structured, and used the questions reproduced below as a guide, although the sequence was in some (rare) cases not strictly followed when interviewees anticipated a question. This type of interview is known as a *guided interview* in the literature (e.g. Patton, 2001; Flick, 1999): the questions are used to control the flow of discussion, and ensure comparable information is covered, yet the direction of the answer, and the extent of information that the interviewee wants to include is left open. This improves the balance of the interview—it enables the identification of topics that are of particular individual interest, and prevents to some degree that the interview flow cuts short individually important answers.

The questions guiding the interview (compare Table 24) were designed to first examine the self-conception of the interviewee regarding her role in the development process, identify personal and common goals of development, and probe for conflicts. The second part encouraged the interviewees to elaborate about project management, and the ways in which a vision was identified, closing with a confrontational question about the use of scenarios in the process. The final part of the interview concentrated on the role of values in the development process, specifically the role of simplicity. The personal motivation for

believing in simplicity, the individual understanding of the term, and conflict arising from this conception were examined.

Table 24: Questions guiding the interview

Q1	Ich möchte mit Dir über die Entwicklung der CommSy-Software sprechen. Welche Rolle hast Du im Entwicklungsteam eingenommen? Was war Dein Verhältnis zu anderen Teammitgliedern? *I'd like to talk about the development of the CommSy software with you. Which role did you have in the development team? What was the relationship with other team members like?*
Q2	Was war für Dich das primäre Ziel im CommSy-Kontext? Entwicklungsziel? Software oder etwas Anderes? *What was the primary goal for you in the context of CommSy? Development goal? Software or something else?*
Q3	Welche Funktion erfüllt für Dich CommSy? Wenn es viele sind, war das ein Problem? *What function fulfills CommSy for you? If there are several functions, did that present a problem?*
Q4	Was waren die für Dich wesentlichen Gründe für Benutzerpartizipation? *Please name your vital reasons for enabling user participation.*
Q5	Wie habt Ihr den Überblick behalten, wohin es gehen soll, wie wurde ein gemeinsames Ziel definiert und wie manifestiert? *How did you maintain an overview of the direction of development, and how did you define and manifest a common goal?*
Q6	Wie ist/war das Verhältnis zwischen Deinem primären Ziel und dem gemeinsamen? *Describe the relationship of your primary goal and the common goal.*
Q7	Gab es Kursänderungen und wie explizit wurden diese (gemacht)? *Did development change course, and if, how explicitly was this communicated?*
Q8	Warum wurden keine (Entwicklungs-)Szenarios bei der Entwicklung verwendet? Wie seid Ihr an Nutzungssituationen gelangt? *Why did you use no development scenarios? How did you capture use situations?*
Q9	Gab es Fälle, in denen ihr den Bedarf der Benutzer völlig falsch eingeschätzt habt? Wann / Warum? *Were there examples for misjudging the requirements of users? When and Why?*
Q10	Welche Rolle hat bei CommSy die Entwicklung von Werten gegenüber der Entwicklung von Software gespielt? Gab es hier Veränderungen (Entwicklungen)? Wichtige Werte? *Which role in the CommSy development did the development of values play vs. the development of software? Were there changes (developments)? Important values?*
Q11	Welche Rolle hat(te) „Einfachheit" bei der Entwicklung von CommSy? (Erst hier!) *Which role has/had „simplicity" for the development of CommSy? (Not before!)*
Q12	Was hat Dich vom Wert „Einfachheit" überzeugt? *What convinced you of the „simplicity" value?*
Q13	Gab es Konflikte, die aus dem Verfolgen des Wertes Einfachheit entstanden sind? Wie wurden diese gelöst? *Were there conflicts resulting from following the value of simplicity? How were they solved?*
Q14	Gibt es Stellen, an denen CommSy zu einfach wurde/ist? Wer findet das? *Are there areas of CommSy that are/have become too simple? Who says so?*

Five interviews were conducted with five members of the core development team. Each interview lasted 47-62 minutes, and notes were taken during the interview. A digital audio recording of the interviews was made for later reference and transcribed verbally.

The results of the five interviews is summarized in Table 25 and Table 26 below, answers are subsumed under category names that were found after comparing all answers. In some cases, the freedom allowed by the form of the interview resulted in answers being given in advance; this is omitted in the tabular form. The complete transcript of the interviews follows after the summary; it is reproduced in the original German, all references to real persons have been replaced by placeholders.

Table 25: Interview analysis (1): function, participation, vision, course changes.

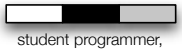
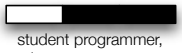
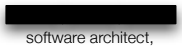
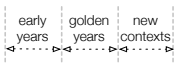
Regarding the *function of CommSy* for themselves, almost all developers considered the software a tool as well as a product, with the programmers adding a focus on technology, and the three current members of the development team also naming monetary support.

The motivation for *implementing participatory design techniques* stem from an understanding of the user as an important role in development, and have been manifested in a project tradition that all developers repeatedly refer to. A high software quality through incremental optimization and interest in unknown contexts are further mentions.

A vision for the project was perceived as being attached to single strong players, with several initial vision slowly converging into the product. Workshops and repeated discussions played an important role in forming that vision; the design principles served as a medium to codify the internal consensus.

All interviewees saw a rather continuous development from the Early Years and through the Golden Years, when the most important customers were the developers themselves. The financial obligations created by the founding of a service consultancy, and the involvement with different contexts created more changes in the last years.

Table 26: Interview analysis (2): values, role and benefits of simplicity, conflicts.

	Membership & Role in the Dev Team	Value Development	Role of Simplicity	Benefits of Simplicity	Conflicts / Too Simple
IV1	 student programmer, concept consultant	gradual changes **simplicity no cure** **personal responsibility crucial**	**learnability** **transparency** graphical	**great wording**	placing of textual labels **growing user base** **new features simplify use (network navigation)**
IV2	 student programmer, primary user contact, lead programmer	**internal motivation (g.y.)** basis for consensus design rationale	most tangible value **functional reduction** communication with users	user acceptance **value adoption by users** criterion for usability	**only partial support for contextual complexities** aspects got simpler, but never too simple **operationalization unclear**
IV3	 evaluation, knowledge archive programmer, user support, awareness advocate	identification with values selected team members **change through contextual change**	made CommSy stand out **central measurement**	**user acceptance** comparison with other systems that lack transparency / consistency	repeatedly became not simple **new features simplify use (awareness, archive)** **operationalization unclear**
IV4	 software architect, lead programmer, user support, release manager, entrepreneur	defined development corridor **change through contextual change**	central measurement learnability	user acceptance even from non-CS users	missing features -> interfaces difficulties selling simplicity **operationalization unclear**
IV5	 project manager	**defined direction of design**	**basis for discussions** understandability	**cure for frustration** transparency **focus on core**	**not simple anymore** frequent conflicts **operationalization unclear**
	 early years / golden years / new contexts				

Values were considered to take a strong role in development: They defined goals for design, and thus a corridor for development. They were a strong internal motivation for the team, and identification with values even became a selection criterion for team members. Other values were considered important as simplicity alone was no cure. Values changes during the development, mostly though changes in use contexts, and the involved users.

Simplicity had a central role among the design values, as it was considered both a central measurement for design success (and confounded with ease of use), and a basis for most discussions. It was seen both as the most tangible value, and as a value that made CommSy stand out from its competitors. The main interpretations of simplicity touched upon learnability, understandability / transparence, and graphical simplicity.

For all interviewees, simplicity was a strongly positive value. Most understood the high user acceptance to be a direct consequence of simplicity in design, especially as users form other disciplines, and with little computer knowledge were quick to adopt CommSy. The value itself was readily adopted by users, and developed from an initial reaction to frustration with other software to a focus on the core of design. Relating to the graphical simplicity, it was connected with great wording: CommSy initially evaded the use of icons, emphasizing the importance of correct wording.

Simplicity was also cause for conflicts, mainly due to its unclear definition. Its operationalization was unclear, and it was repeatedly used by both opposing parties in design

discussions. A current approach is to add features that are missing in CommSy due to the focus on simplicity using open interfaces. CommSy was either considered to be not simple anymore (in the functional sense), or it was thought that it repeatedly became less simple, and then improved again, sometimes through the addition of features, e.g. the network navigation.

9 Bibliography

Albers, J. (1963). *Interaction of color.* New Haven and London: Yale University Press.

Albers, J. & Weber, N. F. (1988). *Josef Albers : a retrospective.* New York: Solomon R. Guggenheim Foundation.

Alexander, C. (1977). *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series).* Oxford University Press, USA.

Alley, M. & Neeley, K. A. (2005). Rethinking the design of presentation slides: a case for sentence headlines and visual evidence. *Technical Communication*, *52*(4), 417-426.

Alloway, L. (1966). Systemic Painting. Introductory essay from the exhibition catalogue Systemic Painting, Solomon R. Guggenheim Foundation, New York. In G. Battcock (Ed.), *Minimal art: a critical anthology. 1995.* (pp. 37-60). Berkeley, Los Angeles and London: University of California Press.

Ambler, S. W. & Jeffries, R. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process.* Wiley.

Anderson, J. R. (1994). *Learning and Memory: An Integrated Approach.* John Wiley and Sons (WIE).

Anderson, R. I. (2000). Conversations with Clement Mok and Jakob Nielsen, and with Bill Buxton and Clifford Nass. *interactions*, *7*(1), 46-80.

Antonio, E. d. (1981). Interview between Emile de Antonio and Frank Stella. *GEO Magazine*.

Apple. (2005). Apple Introduces iPod shuffle: First iPod Under $100. Retrieved 20.7.2006, from http://www.apple.com/pr/library/2005/jan/11shuffle.html

Apple. (2001). Apple Presents iPod. *Press Release* Retrieved 20.7.2006, from http://www.apple.com/pr/library/2001/oct/23ipod.html

Apple. (2006a). Apple Human Interface Guidelines. Retrieved 2.8.2006, from http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf

Apple. (1992). *Macintosh Human Interface Guidelines (Apple Technical Library).* Addison-Wesley Professional.

Apple. (2006b). Mail. You've got more. Retrieved 8.8.2006, from http://www.apple.com/macosx/leopard/mail.html

Apple. (2006c). Time Machine. A giant leap forward for backup. Retrieved 7.8.2006, from http://www.apple.com/macosx/leopard/timemachine.html

Inc., A. C. (2002). Apple Acquires Emagic. Retrieved 17.7.2006, from http://www.apple.com/pr/library/2002/jul/01emagic.html

Ashton, D. (1982). *American Art Since 1945*. Oxford University Press, USA.

Aynsley, J. (2001). *A Century of Graphic Design*. Barron's Educational Series.

Balzert, H. (1985). Allgemeine Prinzipien des Software Engineering. *Angewandte Informatik, 1*, 1-8.

Balzert, H. (2000). *Lehrbuch der Software- Technik 1/2. mit 3 CD-ROMs. Band 1 (2. Auflage, 2000), Band 2 (1. Auflage, 1998) Software- Entwicklung / Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag.

Barreau, D. & Nardi, B. A. (1995). Finding and reminding: file organization from the desktop. *SIGCHI Bull, 27*(3), 39-43.

Barth, J. (1986, 28 December). A Few Words About Minimalism. *The New York Times Book Review*, pp. 2,25.

Barthelme, F. (1988, 3 April). On Being Wrong: Convicted Minimalist Spills Bean [sic]. *New York Times Book Review*, pp. 25-27.

Bass, L., Clements, P., & Kazman, R. (1998). *Software architecture in practice*. Reading, Mass: Addison-Wesley.

Batista, E. (2001). Sex and the Cell Phone-Deprived. *Wired News*. Retrieved 13.10.2006, from http://www.wired.com/news/business/0,1367,48008,00.html

Battcock, G. (1968). *Minimal art; a critical anthology* ([1st ed.] ed.). New York: E. P. Dutton.

Baumert, J. (2004). *StarOffice 4 Kids – Mitwachsende Software im Einsatz*. Mensch & Computer 2004, Allgegenwärtige Interaktion, München, 199-208.

Baumert, J. & Meiners, F. (2003a). Auswertung des Probeeinsatzes von „StarOffice 4 Kids". Retrieved 13.10.2006, from http://www.joachimbaumert.de/downloads/Auswertung_Probeeinsatz.pdf

Baumert, J. & Meiners, F. (2003b). *StarOffice 4 Kids – Mitwachsende Software für den Grundschulunterricht*. Mensch & Computer 2003, Interaktion in Bewegung, Stuttgart, 385-386.

Baumgarten, A. (1983). Kollegium über die Ästhetik, §1. In H. R. Schweizer (Ed.), *Texte zur Grundlegung der Ästhetik*. (pp. 78-83). Hamburg: Felix Meiner.

Bay, S. & Ziefle, M. (2003). Performance in mobile phones: does it depend on a proper cognitive mapping?. In D. e. a. Harris (Ed.), *Human Centred Computing*. (pp. 170-174). Lawrence Erlbaum.

BBC. (2004). iPod helps Apple triple profits. Retrieved 20.7,.2006, from http://news.bbc.co.uk/2/hi/business/3627425.stm

BBC. (2000). Sticking around - the Post-it note is 20. Retrieved 10.10.2006, from http://news.bbc.co.uk/1/hi/uk/701661.stm

Beaudouin-Lafon, M. (2000). *Instrumental interaction: an interaction model for designing post-WIMP user interfaces*. CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 446-453.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison Wesley.

Beck, K. & Andres, C. (2004). *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Agile Manifesto. Retrieved 2.7.2006, from www.agilemanifesto.org

Becker-Pechau, P., Breitling, H., Lippert, M., & Schmolitzky, A. (2003). *Teaching Team Work: An Extreme Week for First-Year Programmers*. Extreme Programming and Agile Processes in Software Engineering, 4th International Conference, XP 2003, 386-393.

Bell, M., Chalmers, M., Barkhuus, L., Hall, M., Sherwood, S., Tennent, P., Brown, B., Rowl, D., & Benford, a. S. (2006). *Interweaving mobile games with everyday life*. CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA, 417-426.

Bellotti, V., Ducheneaut, N., Howard, M., & Smith, I. (2003). *Taking email to task: the design and evaluation of a task management centered email tool*. CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 345-352.

Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., & Neuwirth, C. (2002). *Innovation in extremis: evolving an application for the critical work of email and information management*. DIS '02: Proceedings of the conference on Designing interactive systems, New York, NY, USA, 181-192.

Bellotti, V. & Smith, I. (2000). *Informing the design of an information management system with iterative fieldwork*. DIS '00: Proceedings of the conference on Designing interactive systems, New York, NY, USA, 227-237.

Benson, C., Elman, A., Nickell, S., & Robertson, C. Z. (2004). GNOME Human Interface

Guidelines 2.0.

Bentley, R., Horstmann, T., & Trevor, J. (1997). The World Wide Web as enabling technology for CSCW: The case of BSCW. *Computer-Supported Cooperative Work: Special issue on CSCW and the Web, 6*.

Beocentral. (2005). Beo 4. Retrieved 10.10.2006, from http://www.beocentral.com/products/beo4

Berghel, H. (1997a). Cyberspace 2000: dealing with information overload. *Commun. ACM, 40*(2), 19-24.

Berghel, H. (1997b). Email - the good, the bad, and the ugly. *Commun. ACM, 40*(4), 11-15.

Bergman, E. & Haitani, R. (2000). *Designing the PalmPilot: A Conversation with Rob Haitani. In Bergman: Information Appliances and Beyond*. Morgan Kaufmann.

Bergman, O., Beyth-Marom, R., & Nachmias, R. (2003). The user-subjective approach to personal information management systems. *J. Am. Soc. Inf. Sci. Technol, 54*(9), 872-878.

Berkun, S. (2002). #22 - The list of reasons ease of use doesn't happen on engineering projects. Retrieved 13.10.2006, from http://www.scottberkun.com/essays/essay22.htm

Berkun, S. (2005). #46 - Why software sucks (And what to do about it). Retrieved 13.8.2006, from http://www.scottberkun.com/essays/essay46.htm

Bernard, J. W. (1993). The Minimalist Aesthetic in the Plastic Arts and in Music. *Perspectives of New Music*, *31*(1), 86-132.

Bernard, P., Hammond, N., Morton, J., Long, J., & Clark, I. (1981). Consistency and Compatibility in Human-Computer Dialogue. *International Journal of Man-Machine Studies*, *15*(1), 87-134.

Berners-Lee, T. & Fischetti, M. (1999). *Weaving the Web : the original design and ultimate destiny of the World Wide Web by its inventor* (1st ed ed.). San Francisco: HarperSanFrancisco.

Berners-Lee, T. J. (1989). Information Management: A Proposal.

Bertelsen, O. W. (2006). Tertiary Artifacts at the Interface. In P. Fishwick (Ed.), *Aesthetic computing*. (pp. 357-368). Cambridge, Mass: MIT Press.

Bertelsen, O. W. & Pold, S.,ren (2004). *Criticism as an approach to interface aesthetics*. NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction, New York, NY, USA, 23-32.

Bertoni, F. (2002). *Minimalist Architecture*. Birlhauser (Princeton Architectural Press).

Bevan, N. (1995). *Human-computer interaction standards*. Proceedings of HCI International 95, Volume 2. Advances in Human Factors/Ergonomics, Volume 20B, Amsterdam, The Netherlands, 885-890.

Bevan, N. (2001). International standards for HCI and usability. *Int. J. Hum.-Comput. Stud*, *55*(4), 533-552.

Beyer, H. & Holtzblatt, K. (1997). *Contextual Design : A Customer-Centered Approach to Systems Designs (The Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann.

Beyer, H. R. & Holtzblatt, K. (1995). Apprenticing with the customer. *Commun. ACM*, *38*(5), 45-52.

Beyer, H. R., Holtzblatt, K., & Baker, L. (2004). *An Agile Customer-Centered Method: Rapid Contextual Design*. Extreme Programming and Agile Methods - XP/Agile Universe 2004, 50-59. Available online at http://www.incontextdesign.com/resource/pdf/XPUniverse2004.pdf

Bickford, P. (1997). *Interface Design: the Art of Developing Easy-to-Use Software*. Morgan Kaufmann Pub.

Birk, A., Kohler, K., & Leidermann, F. (2003). *Der Weg zu einer stärkeren Verzahnung von Usability Engineering und Software Engineering*. UPA-Workshop Mensch & Computer 2003.

Birkerts, S. (1986). The School of Lish. *The New Republic*, pp. 28-33.

Bittel, L. R. & Bittel, M. A. (1978). *Encyclopedia of professional management*. New York: McGraw-Hill.

Blood, R. (2002). *The Weblog Handbook: Practical Advice on Creating and Maintaining Your Blog*. Perseus Books Group.

Blood, R. (2000). Weblogs: A History and Perspective. *Rebecca's Pocket* Retrieved 20.7.2006, from http://www.rebeccablood.net/essays/weblog_history.html

Boardman, R. (2004). *Improving Tool Support for Personal Information Management*. Imperial College, London. Available online at http://www.iis.ee.ic.ac.uk/~rick/thesis/boardman04-thesis.pdf

Boardman, R. & Sasse, M. A. (2004). *Stuff goes into the computer and doesn't come out": a cross-tool study of personal information management*. CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 583-590.

Bobrow, D. G., Burchfield, J. D., Murphy, D. L., & Tomlinson, R. S. (1972). TENEX, A Paged Time Sharing System for the PDP-10. *Communications of the ACM*, *15*, 135-143.

Bobzin, A. (2004). *Das Bauhaus-Schachspiel von Josef Hartwig : Entstehungs- und Rezeptionsgeschichte*. Fachhochschule Berlin, Berlin.

Boddie, J. (2005). Has Apple Hit the Right Disruptive Notes? *HSBP Newsletter: Strategy and Innovation*, *3*(4).

Böhr, C. (1985). *Liberalismus und Minimalismus*. Decker R. Von.

Boorstein, J. (1992). *The Hollywood Eye: What Makes Movies Work*. Perennial.

Bouvin, N. O. (1999). *Unifying strategies for Web augmentation*. HYPERTEXT '99: Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots, New York, NY, USA, 91-100.

Bødker, S. (1990). *Through the Interface – a Human Activity Approach to User Interface Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Bødker, S. & Christiansen, E. (1997). Scenarios as springboards in the design of CSCW. In B. e. al. (Ed.), *Social science research, technical systems and cooperative*. (pp. 217-234). Mahwah, NJ: Erlbaum.

Bødker, S. & Christiansen, E. (2004). *Designing for ephemerality and prototypicality*. DIS '04: Proceedings of the 2004 conference on Designing interactive systems, New York, NY, USA, 255-260.

Bødker, S., Grønbæk, K., & Kyng, M. (1995). Cooperative Design: Techniques and Experiences from the Scandinavian Scene. *Human-computer interaction: toward the year 2000*. (pp. 215 - 224). Morgan Kaufmann.

Braa, J. (1996). *Community-based Participatory Design in the Third World*. Proc. PDC'96, 15-23.

Braa, K. (1995). *Priority Workshops: Springboard for User Participation in Redesign Activities*. Proc. COOCS 1995, New York, NY, USA, 246-255.

Bradley-Hole, C. (1999). *The Minimalist Garden*. Monacelli.

Brand, S. (1995). *How Buildings Learn : What Happens After They're Built*. Penguin (Non-Classics).

Brandt, E. & Messeter, J.,rn (2004). *Facilitating collaboration through design games*. PDC 04: Proceedings of the eighth conference on Participatory design, New York, NY, USA, 121-131.

Breen, C. (2006, April). GarageBand 3. *Macworld*.

Breitling, H., Lilienthal, C., Lippert, M., & Züllighoven, H. (2000). *The JWAM Framework: Inspired By Research, Reality-Tested By Commercial Utilization*. Proceedings of OOPSLA 2000 Workshop: Methods and Tools for Object-Oriented Framework Development and Specialization.

Britannica. (2003). *Encyclopedia Britannica: With 2004 Book of the Year (Britannica Books)*. Encyclopedia Britannica Corporation.

Brodie, M. L., Mylopoulos, J., & Schmidt, J. W. (1984). *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages (Topics in Information Systems)*. Springer-Verlag.

Brooks Jr., F. P. (1987). No Silver Bullet - Essence and Accidents of Software Engineering. *IEEE Computer*, *20*(4), 10-19.

Brooks, M. (1999). Introducing the Dvorak Keyboard - Dissenting Opinions.

Brown, B. A. T. & Sellen, A. (2001). Exploring Users' Experiences of the Web. *First Monday*, *6*(9).

Brown, C. M. (1988). *Human-Computer Interface Design Guidelines (Human/Computer Interaction, No 5)*. Ablex Pub.

Brown, D. (2002). Understanding PowerPoint: Special Deliverable #5. *Boxes and Arrows* Retrieved 13.6.2006, from http://www.boxesandarrows.com/view/understanding_powerpoint_special_deliverable_5

Brown, J. S. & Duguid, P. (1991). Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning, and Innovation. Organization. *Science*, *2*(1), 40-57.

Brown, J. S. (1983). *When user hits machine, or When is artificial ignorance better than artificial intelligence?* CHI'83 plenary address, Boston, Massachusetts,.

Brown, J. S. (1990). Research that reinvents the corporation. *Harvard Bus. Rev*, *68*(1), 102.

Bryant, S. L., Forte, A., & Bruckman, A. (2005). *Becoming Wikipedian: transformation of participation in a collaborative online encyclopedia*. GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, New York, NY, USA, 1-10.

BSCW. BSCW Client Extensions for Windows. Retrieved 17.7.2006, from http://bscw.fit.fraunhofer.de/clientex.html

Bürdek, B. E. (1991). *Design. Geschichte, Theorie und Praxis der Produktgestaltung.* DuMont Reiseverlag, Ostfildern.

Bürdek, B. E. (2005). *Design. Geschichte, Theorie und Praxis der Produktgestaltung.* Birkhäuser.

Burrows, P. (1997, July 28). IS APPLE MINCEMEAT? *BusinessWeek*.

Bush, V. (1946). As we may think. Reprinted in 1996. *interactions*, *3*(2), 35-46.

Buskirk, E. v. (2004). Perspective: The secret of iPod's scroll wheel. Retrieved 18.7.2006.2006, from http://news.com.com/The+secret+of+iPod's+scroll+wheel/2010-1041_3-5375101.html

Butler, K. A. (1996). Usability engineering turns 10. *interactions*, *3*(1), 58-75.

Butter, A. & Pogue, D. (2002). *Piloting Palm: The Inside Story of Palm, Handspring, and the Birth of the Billion-Dollar Handheld Industry*. Wiley & Sons.

Button, G. & Dourish, P. (1996). *Technomethodology: paradoxes and possibilities*. CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 19-26.

Buxton, B. (2003). *Performance by Design: The Role of Design in Software Product Development*. Proc. of the Second International Conference on Usage-Centered Design, Portsmouth, NH, 1-15.

Byrne, D. (2003). Learning to Love PowerPoint. *Wired Magazine*, *11.09*.

Byrne, M. D., John, B. E., Wehrle, N. S., & Crow, D. C. (1999). *The tangled Web we wove: a taskonomy of WWW use*. CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 544-551.

Cage, J. (1961). *John Cage: Silence. Lectures and writings by J.C.* Middletown, Conn.: Wesleyan University Press.

Cage, J. (1991). An autobiographical statement. *Southwest Review*, *76*(1), 59.

Cailliau, R. & Ashman, H. (1999). Hypertext in the Web — a history. *ACM Comput. Surv*, *31*(4es), 35.

Calhoun, T. (2005). Bravo for the Duke iPod Experiment. *Campus Technology*.

Calvert, J. B. (2004). The Electromagnetic Telegraph. Retrieved 2.3.2006, from http://www.du.edu/~jcalvert/tel/morse/morse.htm

Cameron, A. & Limited, S. D. (2004). *IdN Special 04 - The Art of Experimental Interaction Design*. Gingko Press.

Campbell, C. S. & Maglio, P. P. (1999). Facilitating navigation in information spaces: road-signs on the World Wide Web. *Int. J. Hum.-Comput. Stud*, *50*(4), 309-327.

Card, S. K. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.

Carliner, S., Dillon, A., Garrett, J. J., Haller, T., Jacobson, B., Quesenberry, W., Shedroff, N., Sless, D., Wodtke, C., Schriver, K., Rosenfeld, L., & Wurman, R. S. (2001). What's in a name? *Design Matters*, *4*, 1-10.

Carlton, J. & Kawasaki, G. (1997). *Apple: The Inside Story of Intrigue, Egomania, and Business Blunders*. Diane Pub Co.

Carr, L. A., {DeRoure, D. C., Hall, W., & Hill, G. J. (1995). *The Distributed Link Service: A Tool for Publishers, Authors and Readers*. Fourth International World Wide Web Conference: The Web Revolution, (Boston, Massachusetts, USA), 647-656.

Carr, N. (2005). The amorality of Web 2.0. Retrieved 28.8.2006, from http://www.roughtype.com/archives/2005/10/the_amorality_0.php

Carroll, J. M. (1990). *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill (Technical Communication, Multimedia, and Information Systems)*. The MIT Press.

Carroll, J. M. (1995a). *Scenario-Based Design: Envisioning Work and Technology in System Development*. John Wiley & Sons.

Carroll, J. M. (1995b). The Scenario Perspective on System Development. In J. M. Carroll (Ed.), *Scenario-Based Design: Envisioning Work and Technology in System Development*. (pp. 1-17). John Wiley and Sons.

Carroll, J. M. (1997). *Reconstructing minimalism*. SIGDOC '97: Proceedings of the 15th annual international conference on Computer documentation, New York, NY, USA, 27-34.

Carroll, J. M. (2002). *Scenarios and Design Cognition*. IEEE Joint International Conference on Requirements Engineering (RE'02), Essen, 3. Available online at http://doi.ieeecomputersociety.org/10.1109/ICRE.2002.1048498

Carroll, J. M. (1998). *Minimalism Beyond the Nurnberg Funnel (Technical Communication, Multimedia, and Information Systems)*. The MIT Press.

Carroll, J. M., Chin, G., Rosson, M. B., & Neale, D. C. (2000). *The development of cooperation: five years of participatory design in the virtual school*. DIS '00: Proceedings of the conference on Designing interactive systems, New York, NY, USA, 239-251.

Carroll, J. M. & Rosson, M. B. (1992). Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Trans. Inf. Syst*, *10*(2), 181-212.

Carroll, J. M., Rosson, M. B., Jr, George Chin, & Koenemann, J. (1998). Requirements Development in Scenario-Based Design. *IEEE Trans. Softw. Eng*, *24*(12), 1156-1170.

Carroll, J. M., Singley, M. K., & Rosson, M. (1992). Integrating theory development with design evaluation. *Behavior & Information Technology*, *11*, 247-255.

Castillo, E. (2004). *Minimalism DesignSource (DesignSource)*. Collins Design.

Chageux, J.-P. & Connes, A. (1995). *Conversations on Mind, Matter and Mathematics*. Princeton, N. J.: Princeton University Press.

Chalmers, M., Bell, M., Hall, M., Sherwood, S., & Tennent, P. (2004). Seamful Games. Retrieved 20.7.2006, from http://www.ubicomp.org/ubicomp2004/adjunct/demos/chalmers.pdf

Chalmers, M. & Galani, A. (2004). *Seamful interweaving: heterogeneity in the theory and design of interactive systems*. DIS '04: Proceedings of the 2004 conference on Designing interactive systems, New York, NY, USA, 243-252.

Chimera, R. & Shneiderman, B. (1993). User Interface Consistency: An Evaluation of Original and Revised Interfaces for a Videodisk Library. In B. Shneiderman (Ed.), *Sparks of Innovation in Human-Computer Interaction*. (pp. 259-273). Norwood, NJ: Ablex Publishers.

Chin, G. J., Rosson, M. B., & Carroll, J. M. (1997). *Participatory analysis: shared development of requirements from scenarios*. CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 162-169.

Chin, J. P. (1987). *Top-down and bottom-up menu design*. Proceedings of the Second International Conference on Human/Computer Interaction, Honululu, HI, 144-147.

Chomsky, N. (1968). *Language and mind*. New York: Harcourt, Brace & World.

Chomsky, N. (1995). *The minimalist program*. Cambridge, Mass: The MIT Press.

Chomsky, N. (1997). Language and Mind: Current Thoughts on Ancient Problems (Part 1). *Pesquisa Lingüística*, *3*(4).

Chomsky, N., Belletti, A., & Rizzi, L. (2002). *On nature and language*. Cambridge: Cambridge University Press.

Ciffolilli, A. (2003). Phantom authority, self-selective recruitment, and retention of members in virtual communities: The case of Wikipedia. *First Monday*, *8*(12).

Clements, P. C. (1999). *Constructing Superior Software (Software Quality Institute Series)*. Sams.

Coad, P., Lefevre, E., & DeLuca, E. (1999). *Java Modeling in Color with UML: Enterprise Components and Process*. Prentice Hall.

Cockton, G. (2004). *Value-centred HCI*. NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction, New York, NY, USA, 149-160.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison Wesley.

Colpitt, F. (1990). *Minimal Art: The Critical Perspective*. Ann Arbor: UMI Research Press.

CommSy. (2005). BenutzerInnenhandbuch. Kontext: Hochschule. Retrieved 10.10.2006, from http://www.commsy.de/downloads/commsy_nutzungshandbuch.pdf

Computer, I. A. (1987). *Apple Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley (C).

Conklin, E. J. (1987). Hypertext: An introduction and survey. *IEEE Computer*, *20*(9), 17-41.

Constantine, L. L. & Lockwood, L. A. D. (1999). *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design (Acm Press Series)*. Addison-Wesley Professional.

Cooper, A. (1999). *The Inmates Are Running the Asylum : Why High Tech Products Drive Us Crazy and How To Restore The Sanity*. Sams.

Cooper, A. (1996). Why I am called "the Father of Visual Basic". Retrieved 20.4.2006, from http://www.cooper.com/alan/father_of_vb.html

Cooper, A. & Reimann, R. M. (2003). *About Face 2.0: The Essentials of Interaction Design*. Wiley.

Coplien, J. O. (1999). Re-evaluating the Architectural Metaphor: Towards Piecemeal Growth. Guest editor introduction to IEEE Software Special Issue. *IEEE Software Special Issue on Architecture Design*, *16*(5), 40-44.

Cornell, B., Dinda, P. A., & Bustamante, F. E. (2004). *Wayback: A User-level Versioning File System for Linux*. The 2004 USENIX Annual Technical Conference, FREENIX track.

Cornell, T. (1997). *Representational Minimalism*. SFB 340 Universität Tübingen.

Coy, W. (1997). Defining Discipline. In C. Freksa, M. Jantzen, & R. Valk (Eds.), *Foundations of Computer Science*. Berlin-Heidelberg-New York: Springer.

Cristol, S. M. & Sealey, P. (2000). *Simplicity Marketing: End Brand Complexity, Clutter, and Confusion*. Free Press.

Critchley, S. (2005). I Want Less And I'm Willing To Pay For It. Retrieved 22.5.2006, from http://www.oreillynet.com/digitalmedia/blog/2005/03/i_want_less_and_im_willing_to.html

Csikszentmihalyi, M. (1991). *Flow: The Psychology of Optimal Experience*. Harper Perennial.

Cuito, A. (2002a). *Minimalist Lofts*. Loft & Hbi.

Cuito, A. (2001). *Minimalist Spaces: Housing/Commercial Spaces/Offices and Public Buildings*. HBI.

Cuito, A. (2002b). *From Minimalism to Maximalism*. Loft Publications S.L. and Hbi.

Cummings, P. (1974). Interview with Sol LeWitt.

Cunningham, W. (2006). Portland Pattern Repository. Retrieved 25.5.2006, from http://c2.com/cgi/wiki?WelcomeVisitors

Curbow, D. (1998a). CHI'98 Xerox Star demo.

Curbow, D. (1998b). Final Xerox Star demo Palo Alto, CA.

Dahm, M., Felken, C., Klein-Bösing, M., Rompel, G., & Stroick, R. (2005). Zur Gebrauchstauglichkeit von Handys – Breite Untersuchung und aktueller Stand. *i-com*, *4*(1), 26-33.

Danis, C., Kellogg, W. A., Lau, T., Dredze, M., Stylos, J., & Kushmerick, N. (2005). *Managers' email: beyond tasks and to-dos*. CHI '05: CHI '05 extended abstracts on Human factors in computing systems, New York, NY, USA, 1324-1327.

Davis, A. M. (1995). *201 Principles of Software Development*. McGraw-Hill.

Davis, P. J., Hersh, R., Marchisotto, E. A., & Rota, G.-C. (1995). *The Mathematical Experience*. Birkhäuser Verlag.

de Luca, J. (1998). Original FDD processes. Retrieved 2.7.2006, from http://www.nebulon.com/articles/fdd/originalprocesses.html

de Luca, J. (2002). Latest FDD processes. Retrieved 2.7.2006, from http://www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

de.internet.com. (2004). Deutsche Konsumenten fühlen sich beim Kauf von Handys und Notebooks überfordert. Retrieved 20.6.2006, from http://de.internet.com/index.php?id=2029134&section=Marketing-Statistics

DeBoer, C. (2004). Harmony Remote Control Simulator, Tech Support & Conclusion. Retrieved 10.10.2006, from http://www.audioholics.com/productreviews/avhardware/HarmonyH688remotecontrol_5.php

Denning, P. J. (1982). ACM president's letter: electronic junk. *Commun. ACM*, *25*(3), 163-165.

Désilets, A., Paquet, S., & Vinson, N. G. (2005). *Are wikis usable?* WikiSym '05: Proceedings of the 2005 international symposium on Wikis, New York, NY, USA, 3-15.

Deutschman, A. (2001). *The Second Coming of Steve Jobs*. Broadway.

dezwozhere. (2003). On a minimalist approach to design. Retrieved 10.10.2006, from http://www.dezwozhere.com/blog/archives/000003.html

Diamond, J. (1997). The Curse of QWERTY

O typewriter! Quit your torture!. *DISCOVER*, *18*(4).

Dijkstra, E. W. (1981). The Psychology of the User. Retrieved 13.10.2006, from http://www.cs.utexas.edu/users/EWD/transcriptions/EWD07xx/EWD791.html

Dijkstra, E. W. (1982). How do we tell truths that might hurt? *SIGPLAN Not*, *17*(5), 13-15.

Dijkstra, E. W. (1989). On the cruelty of really teaching of computing science., V32 (1989) 1398-1408. *Communications of the ACM*, *32*(12), 1398-1108.

Dix, A., Ormerod, T., Twidale, M., Sas, C., Gomes da Silva, P., & McKnight, L. (2006). *Why bad ideas are a good idea*. Proceedings of HCIEd.2006-1 Inventivity, Ballina/Killaloe, Ireland,. Available online at http://www.comp.lancs.ac.uk/computing/users/dixa/papers/HCIed2006-badideas/

Dix, A., Finlay, J., & Abowd, G. D. (1997). *Human-computer Interaction*. Prentice Hall.

Djajadiningrat, J. P., Gaver, W. W., & Fres, J. W. (2000). *Interaction relabelling and extreme characters: methods for exploring aesthetic interactions*. DIS '00: Proceedings of the conference on Designing interactive systems, New York, NY, USA, 66-71.

Donovan, J. & Brereton, M. (2004). *Meaning in Movement: A Gestural Design Game*. Extended Abstracts PDC2004 Artful integration: Interweaving Media, Materials and Practices, Toronto, Canada.

Dorsey, J. G. (1983). *Word processing in an M.I.S environment*. ACM 83: Proceedings of the 1983 annual conference on Computers : Extending the human resource, New York, NY, USA, 225.

Dreyfuss, H. (1955). *Designing for people* (Reprinted in Allworth Press, 2003 ed.). New York: Simon & Schuster.

Dubinsky, J. (1999). Fifteen ways of looking at minimalism. *SIGDOC Asterisk J. Comput. Doc*, *2*(23), 34-47.

Ducheneaut, N. & Bellotti, V. (2001). E-mail as habitat: an exploration of embedded personal information management. *interactions*, *8*(5), 30-38.

Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., & Robbins, D. C. (2003). *Stuff I've seen: a system for personal information retrieval and re-use*. SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, New York, NY, USA, 72-79.

Eaton, M. M. (1998). Locating the aesthetic. In C. Korsmeyer (Ed.), *Aesthetics: The big questions*. (pp. 84-91). Malden, MA: Blackwell.

eco. (2005). Einfache Bedienung von Handys wichtiger als mehr Funktionen. Retrieved 20.7.2006, from http://www.eco.de/servlet/PB/menu/1541961_l1/index.html

Eco, U. (1986). Function and Sign: Semiotics of Architecture. In M. Gottdiener & A. P. Lagopoulos (Eds.), *The City and the Sign*. New York: Columbia University Press.

Eggli, L., Brüderlin, B. D., & Elber, G. (1995). *Sketching as a solid modeling tool*. Proceedings of the Third ACM Symposium on Solid Modeling and Applications, New York, NY, USA, 313-322.

Enderle, R. (2005). Looking Ahead to CES. Retrieved 10.10.2006, from http://www.technewsworld.com/story/39326.html

Eustice, K. F., Lehman, T. J., Morales, A., Munson, M. C., Edlund, S., & Guillen, M. (1999). A universal information appliance. *IBM Systems Journal*, *38*(4), 575-601.

Farhoomand, A. F. & Drury, D. H. (2002). Managerial information overload. *Commun. ACM*, *45*(10), 127-131.

Fatton, A., Romberger, S., & Eklundh, K. S. (1993). *The paper model for computer-based writing*. CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 514.

Feller, J. & Fitzgerald, B. (2000). *A framework analysis of the open source software development paradigm*. ICIS '00: Proceedings of the twenty first international conference on Information systems, Brisbane, Queensland, Australia, 58-69.

Fenton, N. E. & Pfleeger, S. L. (1996). *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press.

Ferebee, A. (1994). *A History of Design from the Victorian Era to the Present: A Survey of the Modern Style in Architecture, Interior Design, Industrial Design, Graphic Design, and Photography*. Van Nostrand Reinhold/co Wiley.

Finck, M. & Jackewitz, I. (2006). Flyer zur Software CommSy. Retrieved 13.10.2006, from http://service.commsy.net/downloads/commsy_flyer_produkt.pdf

Finck, M. & Janneck, M. (2006). Fallbeispiele der CommSy-Nutzung im Kontext virtueller Organisationen und Netzwerke - Eine Sammlung von Nutzungsberichten.

Finck, M., Janneck, M., Obendorf, H., & Gumm, D. (2006). *CCS – eine Methode zur kontextübergreifenden Softwareentwicklung*. Mensch & Computer 2006: Mensch und Computer im StrukturWandel, München, 93-102.

Finck, M., Obendorf, H., & Pape, B. (2004). Fallbeispiele der CommSy-Nutzung – Eine Sammlung von Nutzungsberichten. *FBI-HH-B-261/04*.

Findlater, L. & McGrenere, J. (2004). *A comparison of static, adaptive, and adaptable menus*. CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 89-96.

Finkelstein, E. (2003). *How to Do Everything with Microsoft Office PowerPoint 2003 (How to Do Everything)*. Mcgraw-Hill Osborne Media.

Fischer, G. (2001). *External and sharable artifacts as sources for social creativity in communities of interest*. Computational and cognitive models of creative design V: Reprints of the Fifth International Roundtable Conference on Computational and Cognitive Models of Creative Design, Sydney, 67-89.

Fischer, G. (2004). *Social creativity: turning barriers into opportunities for collaborative design*. PDC 04: Proceedings of the eighth conference on Participatory design, New York, NY, USA, 152-161.

Fischer, G. (2006). Beyond binary choices: Understanding and exploiting trade–offs to enhance creativity. *First Monday*, *4*(11).

Flach, J. & Dominguez, C. (1995). Use-centered design: Integrating the user, instrument, and goal. *Ergonomics in Design*, *3*(3), 19-24.

Flick, U. (1999). *Qualitative Forschung. Theorie, Methoden, Anwendung in Psychologie und Sozialwissenschaften*. Rowohlt Tb.

Floyd, C. (1987). Outline of a Paradigm Change in Software Engineering. In B. e. al. (Ed.), *Computers and Democracy - A Scandinavian Challenge*.

Floyd, C. (1993). STEPS - A Methodical Approach to PD. *Commun. ACM*, *36*(6), 83.

Floyd, C. (1994). Software-Engineering - und dann? *Informatik Spektrum*, *17*(1).

Floyd, C. & Oberquelle, H. (2003). Softwaretechnik und –ergonomie. Skript zur Vorlesung.

Floyd, C. & Piepenburg, U. (1993). *STEPS - ein softwaretechnischer Projektansatz und seine arbeitswissenschaftliche Begründung*. GI Jahrestagung, 145-150.

Floyd, C., Reisin, F.-M., & Schmidt, G. (1989). *STEPS to Software Development with Users*. ESEC '89, 2nd European Software Engineering Conference, 48-64.

Fogg, B. J. (2002). *Persuasive Technology: Using Computers to Change What We Think and Do (The Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann.

Fowler, M. (2003). *UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition*. Addison-Wesley Professional.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional.

Franklin, J. (1994). *Writing for Story: Craft Secrets of Dramatic Nonfiction*. Plume.

Franzke, M. & Rieman, J. (1993). *Natural Training Wheels: Learning and Transfer Between Two Versions of a Computer Application*. VCHCI '93: Proceedings of the Vienna Conference on Human Computer Interaction, London, UK, 317-328.

Fried, J. (2004). Web 2.0 Review: Don't forget there's another kind of scaling. Retrieved 20.8.2006, from http://www.37signals.com/svn/archives/000881.php

Fried, M. (1998). *Art and objecthood : essays and reviews*. Chicago: University of Chicago Press.

Fried, M. (1966). Shape as Form: Frank Stella's New Paintings. *Artforum*, *5*(3), 18-27.

Gadamer, H. G. (1998). From "Truth and Method". In C. Korsmeyer (Ed.), *Aesthetics: The big questions*. (pp. 91-97). Malden, MA: Blackwell.

Gadomski, K. E. & Kral, K. M. (1990). *Using macros \& style sheets to aid in document layout*. SIGUCCS '90: Proceedings of the 18th annual ACM SIGUCCS conference on User services, New York, NY, USA, 131-136.

Galloway, A., Brucker-Cohen, J., Gaye, L., Goodman, E., & Hill, D. (2004). *Design for hackability*. DIS '04: Proceedings of the 2004 conference on Designing interactive systems, New York, NY, USA, 363-366.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series)*. Addison-Wesley Professional.

Gantt, M. & Nardi, B. A. (1992). *Gardeners and gurus: patterns of cooperation among CAD users*. CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 107-117.

Gardner, J. (1991). *The Art of Fiction: Notes on Craft for Young Writers (Vintage)*. Vintage.

Garrett, J. J. (2005). Ajax: A New Approach to Web Applications. Retrieved 26.8.2006, from http://www.adaptivepath.com/publications/essays/archives/000385.php

Gartner. (2004). Gartner Says Microsoft's Windows CE Surpassed Palm OS Shipments in Worldwide PDA Operating Systems Market in Third Quarter of 2004. Retrieved 20.7.2006, from http://www.gartner.com/press_releases/asset_113913_11.html

Gedenryd, H. (1998). *How designers work (Lund University cognitive studies)*. Unpublished PhD Dissertation, Lund University, Lund. Available online at http://www.lucs.lu.se/People/Henrik.Gedenryd/HowDesignersWork/

Gelernter, D. (1998). *The Aesthetics of Computing*. Phoenix House.

Gelernter, D. (1999). *Machine Beauty: Elegance and the Heart of Technology (Repr ed) (Masterminds)*. Basic Books.

Gena, P. (1981). Freedom in Experimental Music: The New York Revolution. *Tri-Quarterly*, 52, 223-243.

Gibson, B. (2005). First on TMO - Apple Exec: Shuffle Grabs 58% of Flash Player Market; What Cell Phone Threat? Retrieved 2.10.2006, from http://www.macobserver.com/article/2005/05/04.4.shtml

Gibson, J. J. (1979). *Ecological Approach to Visual Perception*. Houghton Mifflin.

Gillies, J. & Cailliau, R. (2000). *How the Web was Born: The Story of the World Wide Web*. Oxford University Press, USA.

Glaser, B. (1966). Questions to Stella and Judd. *Art News*, Reprinted in: Battcock, Gregory (ed.): Minimal art: a critical anthology. University of California Press, Berkeley, Los Angeles and London, 148-164, 1995.

Glenn, W. (2000a). Top 12 Word Tips. *WindowsDevCenter.com* Retrieved 25.5.2006, from http://www.windowsdevcenter.com/pub/a/oreilly/windows/news/word2k_0900.html

Glenn, W. J. (2000b). *Word 2000 in a Nutshell: A Power User's Quick Reference*. O'Reilly Media, Inc.

Glinert, S. (2006). Google SketchUp. *CNET Editors review* Retrieved 28.8.2006, from http://reviews.cnet.com/Google_SketchUp/4505-3633_7-31861341-2.html

Good, J. V. & Good, P. (1995). Is Functionalism Functional? The Relationship Between Function and Purity. *Communication Arts, 37*(1), 27.

Green, T. R. G. (1989). Cognitive dimensions of notations. In A. Sutcliffe & L. Macaulay(pp. 443-460). Cambridge: Cambridge University Press.

Greenaway, P. & Rauterberg, H. (2005, 29.12.). Rembrandt war kein Maler. *Die Zeit*, p. 41. Zeit Verlag.

Greenbaum, J., Snelling, L., Jolly, C., & On', J. (1994). *The limits of PD? Contingent jobs and work reorganization*. PDC 1994, 173-174.

Greenbaum, J., Kyng, M., & King, M. (1991). *Design at Work: Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates Inc,US.

Greenberg, C. (1962). After Abstract Expressionism. *Art International, VI*(8), 30-32.

Greenwood, W. (2002). Product complexity driving you crazy? Learn where to cut. Retrieved 13.10.2006, from http://www.cooper.com/newsletters/2002_06/product_complexity-learn_where_to_cut.htm

Gropius, W. (1919). The Bauhaus Manifesto. Retrieved 23.7.2006, from http://www.dmoma.org/lobby/Bauhaus_manifesto.html

Grudin, J. (1986). *Designing in the dark: logics that compete with the user*. CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 281-284.

Grudin, J. (1988). *Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces*. CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work, New York, NY, USA, 85-93.

Grudin, J. (1989). The case against user interface consistency. *Commun. ACM, 32*(10), 1164-1173.

Grudin, J. (1990a). *Interface*. CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work, New York, NY, USA, 269-278.

Grudin, J. (1990b). *The computer reaches out: the historical continuity of interface design*. CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 261-268.

Grudin, J. (1992). Consistency, standards, and formal approaches to interface development and evaluation: a note on Wiecha, Bennett, Boies, Gould, and Greene. *ACM Trans. Inf. Syst, 10*(1), 103-111.

Grudin, J. (1994). Groupware and social dynamics: eight challenges for developers. *Commun. ACM, 37*(1), 92-105.

Grudin, J., Tallarico, S., & Counts, S. (2005). *As technophobia disappears: implications for design*. GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, New York, NY, USA, 256-259.

Gundelsweiler, F., Memmel, T., & Reiterer, H. (2004). *Agile Usability Engineering*. Mensch & Computer 2004: Allgegenwärtige Interaktion, München, 33-42.

Gwizdka, J. (2004). *Email task management styles: the cleaners and the keepers*. CHI '04: CHI '04 extended abstracts on Human factors in computing systems, New York, NY, USA, 1235-1238.

Haddad, A. (1999). *Sams Teach Yourself Microsoft PowerPoint 2000 in 24 Hours*. Sams.

Hales, P. (2006). Microsoft wants users to actually use Office 2007. Retrieved 25.5.2006, from http://www.theinquirer.net/?article=31960

Hallett, C. W. (2000). *Minimalism and the Short Story–Raymond Carver, Amy Hempel, and Mary Robison (Studies in Comparative Literature)*. Edwin Mellen Press.

Hammersley, B. (2005). *Developing Feeds with Rss and Atom*. O'Reilly Media.

Hardy, G. H. (1940). *A Mathematician's Apology*. Cambridge University Press.

Hardy, I. R. (1996). *The evolution of ARPANET email*. University of California at Berkeley, Berkeley, CA. Available online at http://www.ifla.org/documents/internet/hari1.txt

Harris, J. (2006a). Grading On the Curve (Why the UI Part 8). Retrieved 10.10.2006, from http://blogs.msdn.com/jensenh/archive/2006/04/11/573348.aspx

Harris, J. (2006b). The End of Personalized Menus. Retrieved 10.10.2006, from http://blogs.msdn.com/jensenh/archive/2006/01/20/515328.aspx

Harris, J. (2005). Tipping the Scale (Why the UI, Part 5). Retrieved 10.10.2006, from http://blogs.msdn.com/jensenh/archive/2005/10/24/484131.aspx

Harris, J. (2006c). Let's Talk About Customization. Retrieved 2006.28.6, from http://blogs.msdn.com/jensenh/archive/2006/06/27/648269.aspx

Harris, J. (2006d). Combating the perception of bloat. Retrieved 2006.28.6, from http://blogs.msdn.com/jensenh/archive/2006/03/31/565877.aspx

Harris, J. (2006e). Inside Deep Thought (Why the UI Part 6). Retrieved 10.10.2006, from http://blogs.msdn.com/jensenh/archive/2006/04/05/568947.aspx

Harris, J. (2006f). No Distaste for Paste (Why the UI Part 7). Retrieved 10.10.2006, from http://blogs.msdn.com/jensenh/archive/2006/04/07/570798.aspx

Harrison, W. (2004). From the Editor: The Dangers of End-User Programming. *IEEE Softw*, *21*(4), 5-7.

Hassenzahl, M. (2004). The Interplay of Beauty, Goodness and Usability in Interactive Products. *uman-Computer Interaction*, *19*, 319-349.

Hassenzahl, M., Burmester, M., & Koller, F. (2003). *AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität*. Mensch & Computer 2003. Interaktion in Bewegung, Stuttgart, 187-196.

Hassenzahl, M., Platz, A., Burmester, M., & Lehner, K. (2000). *Hedonic and ergonomic quality aspects determine a software's appeal*. CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 201-208.

Heckel, P. (1984). *The Elements of Friendly Software Design*. Warner Books.

Heineman, G. & Councill, W. T. (2001). *Component-based Software Engineering: Putting the Pieces Together (ACM Press Books)*. Addison Wesley.

Heller, D. (2005). Aesthetics and interaction design: some preliminary thoughts. *interactions*, *12*(5), 48-50.

A., H. & M., K. (1991). There's no place like home: Continuing Design in Use. In J. Greenbaum & M. Kyng (Eds.), *Design at work: Cooperative Design of Computer Systems*. (pp. 219 - 240). Hillsdale, NJ: Lawrence Erlbaum Ass.

Herder, E., Weinreich, H., Obendorf, H., & Mayer, M. (2006). *Much to Know About History*. AH 2006 4th International Conference for Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland, 283-287.

Hertzum, M. (2003). Making use of scenarios: a field study of conceptual design. *Int. J. Hum.-Comput. Stud*, *58*(2), 215-239.

Hess, T. B. (1963). The Phony Crisis in American Art. *Art News*, *Summer*, 24-28.

Highsmith, J. (2002). *Agile Software Development Ecosystems*. Addison Wesley.

Hightower, R., Ring, L., Helfman, J., Bederson, B., & Hollan, J. (1998). *Graphical multiscale web histories: A study of PadPrints*. ACM Hypertext'98, 58-65. Available online at http://citeseer.ifi.unizh.ch/hightower98graphical.html

Hilgenstock, R. & Jirmann, R. (2006). moodle in Deutschland. Retrieved 19.7.2006, from http://moodle.de/file.php/1/whitepaper.pdf

Hill, D. (2002). Towards designing adaptive systems London: AIGA Experience Design forum.

Hill, J. (1998). *Occupying Architecture*. Spon Press.

Hirschheim, R., Klein, H. K., & Lyytinen, K. (1995). *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations (Cambridge Tracts in Theoretical Computer Science)*. Cambridge University Press.

Hoffman, M. E. & Vance, D. R. (2005). *Computer literacy: what students know and from whom they learned it*. SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education, New York, NY, USA, 356-360.

Hoffmann, H. & Zerche, R. (2005). *Agile Usability Engineering. Entwurf einer agilen Methode zur Entwicklung interaktiver Softwaresysteme unter Berücksichtigung von Usability Engineering.* University of Hamburg, Hamburg.

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Trans. Comput.-Hum. Interact*, *7*(2), 174-196.

Holtzblatt, K., Wendell, J. B., & Wood, S. (2004). *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design (The Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann.

Hood, C. S. & Hood, D. J. (2005). *Teaching programming and language concepts using LEGOs\&\#174;*. ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, New York, NY, USA, 19-23.

Höök, K. (2000). Steps to take before intelligent user interfaces become real. *Interacting with Computers*, *12*(4), 409-426.

Hudson, W. (2005). Playing your cards right: getting the most from card sorting for navigation design. *interactions*, *12*(5), 56-58.

Hughes, J. (2005). Technica – Lego Technic Site: The Automatic Binding Brick. Retrieved 20.8.2006, from http://isodomos.com/technica/history/1940/1949.php

Hughes, R. J., Shewmake, J., & Okelberry, C. R. (1998). *Ceilidh: collaborative writing on the Web*. SAC '98: Proceedings of the 1998 ACM symposium on Applied Computing, New York, NY, USA, 732-736.

Huntley, H. E. (1970). *The divine proportion: a study in mathematical beauty*. New York: Dover Publications.

Hutchings, W. (1986). Abated Drama: Samuel Beckett's Unbated 'Breath'. *Ariel, 17*, 85-94.

Hutchins, E., Hollan, J. D., & Norman, D. A. (1986). Direct Manipulation Interfaces. In D. Norman & S. Draper (Eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction*. (pp. 87-124). Lawrence Erlbaum Associates.

Hutchins, E. (1996). *Cognition in the Wild*. The MIT Press.

IAwiki. (2006). Defining the Damn Thing. Retrieved 3.10.2006, from http://www.iawiki.net/DefiningTheDamnThing

IEEE. (1990). *610.12-1990: IEEE Standard Glossary of Software Engineering Terminology* (610.12). IEEE.

Igarashi, T., Matsuoka, S., & Tanaka, H. (1999). *Teddy: a sketching interface for 3D freeform design*. SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, NY, USA, 409-416.

Inc, N. C. (1992). *Nextstep User Interface Guidelines: Release 3 (Nextstep Developer's Library)*. Addison Wesley Publishing Company.

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., & Kay, A. (1997). *Back to the future: the story of Squeak, a practical Smalltalk written in itself*. OOPSLA '97: Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, NY, USA, 318-326.

Iorio, A. D. & Vitali, F. (2005). *From the writable web to global editability*. HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, New York, NY, USA, 35-45.

ISO9241. (1998). ISO 9241: Ergonomic requirements for office work with visual display terminals (VDTs).

Jackewitz, I. (2005). Evolutionary Application Service Providing - Ein Ansatz der Softwarebereitstellung.

Jackewitz, I., Janneck, M., Krause, D., Pape, B., & Strauss, M. (2003). Teaching Social Informatics as a Knowledge Project. In T. J. van Weert & R. K. Munro (Eds.), *Informatics and the Digital Society*. (pp. 261-268). Kluwer Academic Publishers.

Jackewitz, I., Janneck, M., & Pape, B. (2002). *Vernetzte Projektarbeit mit CommSy*. Mensch & Computer 2002: Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten.

Jackewitz, I., Janneck, M., & Strauss, M. (2004). CommSy: Softwareunterstützung für Wissensprojekte. In B. Pape, D. Krause, & H. Oberquelle (Eds.), *Wissensprojekte – Gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*. (pp. 186-202). Münster: Waxmann.

Jackson, D. (1999). *Victorinox Original Schweizer Offiziersmesser*. Heel.

Janneck, M. (2006a). *Gebrauchstaugliche didaktische Software. Entwicklungsprozess, Didaktik, Gestaltungsprinzipien*. Universität Hamburg.

Janneck, M., Krause, D., Pape, B., & Strauss, M. (2003). *Softwareunterstützung für offene Seminare*. DeLFI 2003, Tagungsband der 1. e-Learning Fachtagung Informatik, 47-56.

Janneck, M. (2006b). *Softwaregestaltung für die Gruppeninteraktion im Kontext von CSCL – am Beispiel der Fallstudie CommSy*. Universität Hamburg,. Available online at http://www.sub.uni-hamburg.de/opus/volltexte/2006/3042/

Janneck, M., Finck, M., & Obendorf, H. (2006a). *Participatory Design: An Issue for Web-Based Community Development?! In: (eds): Web Based Communities*. Proceedings of the IADIS International Conference Web Based Communities 2006, 274-277.

Janneck, M. & Finck, M. (2006). *Appropriation and Mediation of Technology Use in Stable Self-organized Online Communities*. Proc WBC 2006.

Janneck, M., Finck, M., & Obendorf, H. (2006b). *Grenzen bei der Verwendung von Leitbildern - ein Fallbeispiel*. Mensch & Computer 2006: Mensch und Computer im StrukturWandel, München, 73-82.

Janneck, M., Obendorf, H., Finck, M., & Janneck, M. (2006). *Tying Collaborative Knowledge Nets*. Cooperative Systems Design -- COOP 2006, 57-65.

Jarke, M., Bui, X. T., & Carroll, J. M. (1998). Scenario Management: An Interdisciplinary Approach. *Requirements Engineering Journal*, *3*(3-4).

Jaynes, E. T. (1992). The Gibbs Paradox. In C. R. Smith, G. J. Erickson, & P. O. Neudorfer (Eds.), *Maximum Entropy and Bayesian Methods*. (pp. 122). Dordrecht, Holland: Kluwer Academic Publishers.

Jeenicke, M. (2005). *Architecture-Centric Software Migration for the Evolution of Web-based Systems*. Workshop on Architecture-Centric Evolution (ACE 2005), ECOOP 2005, Glasgow,. Available online at http://swt-www.informatik.uni-hamburg.de/publications/download.php?id=322

Jeffries, R., Anderson, A., Hendrickson, C., & Jeffries, R. E. (2000). *Extreme Programming Installed*. Addison-Wesley Professional.

Jeffries, R. (2001). What is Extreme Programming? Retrieved 6.8.2006, from http://www.xprogramming.com/xpmag/whatisxp.htm#whole

Jehn, L. A., Rine, D. C., & Sondak, N. (1978). Computer science and engineering education: Current trends, new dimensions and related professional programs. *SIGCSE Bull*, *10*(3), 162-178.

Jencks, C. (1977). *The language of post-modern architecture* (An Architectural design monograph). London: Academy Editions.

Jensen, B. (2000). *Simplicity: The New Competitive Advantage In A World Of More, Better, Faster.* Diane Pub Co.

Jenson, S. (2002). *The Simplicity Shift: Innovative Design Tactics in a Corporate World*. Cambridge University Press.

Jenson, S. (2005). Default Thinking: Why Mobile Services are setup to fail. In R. Harper (Ed.), *The Inside Text: Social, Cultural and Design Perspectives on SMS*. (pp. 305-325).

Jobs, S. (2001). Apple Music Event 2001-The First Ever iPod Introductio On *Apple Music Event 2001* : Apple Computer.

Jobs, S. (2005). Macworld 2005 Keynote: iPod Shuffle introduction San Francisco: Apple Computers, Inc.

Johnson, J. & Henderson, A. (2002). Conceptual models: begin by designing what to design. *interactions*, *9*(1), 25-32.

Johnson, J. A. & Nardi, B. A. (1996). Creating presentation slides: a study of user preferences for task-specific versus generic application software. *ACM Trans. Comput.-Hum. Interact*, *3*(1), 38-65.

Johnson, P.-A. (1994). *The Theory of Architecture: Concepts, Themes & Practices (Architecture)*. Van Nostrand Reinhold.

Johnson, T. (1991). The Voice of New Music.

Jones, W., Bruce, H., & Dumais, S. (2001). *Keeping found things found on the web*. CIKM '01: Proceedings of the tenth international conference on Information and knowledge management, New York, NY, USA, 119-126.

Judd, D. (1965). Specific Objects. *Arts Yearbook 8*. (pp. 23-35). Reprinted in Donald Judd: Complete Writings 1959-1975 (Halifax: The Press of the Nova Scotia College of Art & Design; and New York: New York University Press, 1975), pp. 181-89.

Judd, D. (1975). *Complete Writings 1959-1975*. New York: New York University Press.

Jun'ichiro, K. (2001). Wapro: the interface between the Japanese language and the computer. *The Book & the Computer. Online journal. Special Series: Japanese in the age of Technology*, *5*(1).

Kahn, H. (1962). *Thinking the unthinkable*. Weidenfeld and Nicolson.

Kane, D. (2003). *Finding a Place for Discount Usability Engineering in Agile Development: Throwing Down the Gauntlet*. ADC '03: Proceedings of the Conference on Agile Development, Washington, DC, USA, 40.

Kang, H., Plaisant, C., & Shneiderman, B. (2003). *New approaches to help users get started with visual interfaces: multi-layered interfaces and integrated initial guidance*. dg.o '03: Proceedings of the 2003 annual national conference on Digital government research, Boston, MA, 1-6.

Karat, J. & Karat, C. M. (2003). The evolution of user-centered focus in the human-computer interaction field. *IBM Syst. J*, *42*(4), 532-541.

Karjalainen, T. M. (2003). *Strategic Brand Identity and Symbolic Design Cues*. 6th Asian Design

Conference, Tsukuba, Japan, 14-17.

Karvonen, K. (2000). *The beauty of simplicity*. CUU '00: Proceedings on the 2000 conference on Universal Usability, New York, NY, USA, 85-90.

Katzen, M. (2006). Delicious Minimalism. *Harvard Magazine*, 31-32.

Kaye, J. J., Vertesi, J., Avery, S., Dafoe, A., David, S., Onaga, L., Rosero, I., & Pinch, T. (2006). *To have and to hold: exploring the personal archive*. CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA, 275-284.

KDE. (2005). KDE 3 Styleguide. Retrieved 20.7.2006, from http://developer.kde.org/documentation/standards/kde/style/styleguide.pdf

Keil-Slawik, R. & Baumert, J. (2004). StarOffice 4 Kids - Mitwachsende Software für den lernenden Nachwuchs. *Forschungs Forum Paderborn*, *7*, 18-22.

Kellogg, W. (1989). The Dimensions of Consistency. In J. Nielsen (Ed.), *Coordinating user interfaces for consistency*. (pp. 9-20). Boston, MA: Academic Press.

Khemlani, L. (2004). SketchUp 4.0. *AECbytes Product Review* Retrieved 31.8.2006, from http://www.aecbytes.com/review/2004/SketchUp4.html

Kiljander, H. (2004). *Evolution and Usability of Mobile Phone Interaction Styles*. Helsinki University of Technology, Espoo, Finland. Available online at http://lib.hut.fi/Diss/2004/isbn9512273209/isbn9512273209.pdf

King, J. P. (1992). *The Art of Mathematics,*. New York: Fawcett Columbine.

Klassner, F. (2002). *A case study of LEGO Mindstorms'\™ suitability for artificial intelligence and robotics courses at the college level*. SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education, New York, NY, USA, 8-12.

Klinkowitz, J. (1993). The New Fiction. In M. Cunliffe (Ed.), *The Penguin History of Literature: American Literature since 1900*. (pp. 353-367).

Klockar, T., Carr, D., Hedman, A., Johansson, T., & Bengtsson, F. (2003). *Usability of mobile phones*. Proceedings of the 19th International Symposium on Human Factors in Telecommunications, Berlin, Germany, 197-204.

Koch, R. (1999). *The 80/20 Principle: The Secret to Success by Achieving More with Less*. Currency.

Koerner, B. I. (2006, February). Geeks in Toyland. *Wired Magazine*.

Koiso-Kanttila, N. (2003). Consumers on the Web: Identification of usage patterns. *First Monday*, *8*(4).

Kopetzky, T. & M\&\#252,hlh\&\#228;user, M. (1999). *Visual preview for link traversal on the World Wide Web*. WWW '99: Proceeding of the eighth international conference on World Wide Web, New York, NY, USA, 1525-1532.

Korpela, M., Soriyan, H. A., Olufokunbi, K. C., Onayade, A. A., Davies-Adetugbo, A., & Adesanmi, D. (1998). Community Participation in Health Informatics inAfrica: An Experiment in Tripartite Partnership inIle-Ife, Nigeria. *Comput. Supported Coop. Work*, *7*(3-4), 339-358.

Kostof, S. (1977). *The Architect: Chapters in the History of the Profession*. Oxford University Press, USA.

Kosuth, J. (1991). *Art after philosophy and after : collected writings, 1966-1990*. Cambridge, Mass: MIT Press.

Kotler, P. (1975). *Marketing for Nonprofit Organizations*. Volunteer Readership.

Koved, L. & Scneiderman, B. (1986). Embedded menus: selecting items in context. *Commun. ACM*, *29*(4), 312-318.

Krauss, R. E. (1977). Passages in Modern Sculpture.

Krauss, R. E. (1991). Overcoming the Limits of Matter: On Revising Minimalism. In J. Elderfield (Ed.), *American Art of the 1960s*. New York: Museum of Modern Art.

Kyng, M. (1995a). Creating contexts for design. *Scenario-based design: envisioning work and technology in system development*. (pp. 85-107). John Wiley \& Sons, Inc. New York, NY, USA.

Kyng, M. (1995b). Making representations work. *Commun. ACM*, *38*(9), 46-55.

Lang, S. (1985). The Beauty of Doing Mathematics.

Laurel, B. (1990). *The Art of Human-Computer Interface Design*. Addison-Wesley Professional.

Laurel, B. (1993). *Computers as Theatre*. Addison-Wesley Professional.

Lave, J., Wenger, E., Pea, R., Brown, J. S., & Heath, C. (1991). *Situated Learning : Legitimate Peripheral Participation (Learning in Doing: Social, Cognitive & Computational Perspectives)*. Cambridge University Press.

Leishman, D. (2004, January). NAMM Wrapup: The logic of NAMM. *Macworld*.

Leuf, B. & Cunningham, W. (2001). *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional.

Levin, K. (1979). Farewell to Modernism. *Arts Magazine*, *54*(2), 90-91.

Levine, R. (1996). Guide to Web Style. Retrieved 10.10.2006, from http://web.archive.org/web/19990508134652/www.sun.com/styleguide/

Levinson, E. (1998). RFC 2387: The MIME Multipart/Related Content-type.

Levy, L. (1996). *Peter Walker Minimalist Gardens*. Whitney Library of Design.

Levy, S. (2006a). Q&A: Jobs on iPod's Cultural Impact. *Newsweek Technology* Retrieved 17.10.2006, from http://msnbc.msn.com/id/15262121/site/newsweek?www.reghardware.co.uk

Levy, S. (2006b). *The Perfect Thing*. Simon & Schuster.

LeWitt, S. (1978a). Paragraphs on Conceptual Art. In A. Legg (Ed.), *Sol LeWitt*. New York: The Museum of Modern Art.

LeWitt, S. (1978b). The Cube. In A. Legg (Ed.), *Sol LeWitt*. New York: The Museum of Modern Art.

Liebowitz, S. J. & Margolis, S. E. (1990). The Fable of the Keys. *Journal of Law & Economics*, *33*.

Lien, J. (2001, 15.10.2001). Is Palm losing the edge? *Business Times*.

Light, A. (2005). Report: Aesthetic Approaches to HCI. Retrieved 20.7.2006, from http://www.usabilitynews.com/news/article2174.asp

Lilienthal, C. & Züllighoven, H. (1997). Application-oriented usage quality: the tools and materials approach. *interactions*, *4*(6), 35-41.

Lindholm, C. & Keinonen, T. (2003). *Mobile Usability: How Nokia Changed the Face of the Mobile Phone*. McGraw-Hill Professional.

Lippard, L. (1967). Sol LeWitt: Non-Visual Structures. *Artforum*, *April 1967*, 42-46.

Lippard, L. (1981). *Ad Reinhardt*. New York: Abrams.

Lipson, H. & Shpitalni, M. (1996). Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-aided Design*, *28*(8), 651-663.

Lithgow, A. (1987, July 26). The Ghost that is Haunting Lego Land. *The Mail on Sunday*.

Lok, L. T. T. (2004). *A Critical Survey of Software Packages for Use by Interior Designers*. Unpublished MSc Thesis, University of Wales, Aberystwyth. Available online at http://www.aber.ac.uk/compsci/Dept/Teaching/MSc_dissertations/2004/Lynda_Lok.pdf#search=%22lynda%20lok%20%20sketchup%20interior%20design%20architecture%22

Lovejoy, T. & Grudin, J. (2003). *Messaging And Formality: Will IM Follow in the Footsteps of Email?* Human Computer Interaction - Interact 2003, 817-820.

Löwgren, J. (1995). *Applying design methodology to software development*. New York, NY, USA, 87-95.

Löwgren, J. (2006). Articulating the use qualities of digital designs. In P. Fishwick (Ed.), *Aesthetic computing*. (pp. 383-403). Cambridge, Mass: MIT Press.

Lynch, P. & Horton, S. (1999). Web Style Guide : Basic Design Principles for Creating Web Sites. Retrieved 10.10.2006, from http://web.archive.org/web/20000816062833/http://www.info.med.yale.edu/caim/manual/, originally at http://info.med.yale.edu/caim/manual/

m-travel. (2002). British business travelers love their mobile phones. *m-travel.com Travel Distribution News*. Retrieved 13.10.2006, from http://www.m-travel.com/news/2002/04/british_busines.html

Macjams.com. (2004a). Interview with GarageBand expert Francis Preve. Retrieved 23.5.2006, from http://www.macjams.com/article.php?story=20041115062201549

Macjams.com. (2004b). GarageBand Tutorial: Built-in Audio Unit Effects. Retrieved 17.7.2006, from http://www.macjams.com/article.php?story=20040329063101758

Mack, R. L., Lewis, C. H., & Carroll, J. M. (1983). Learning to use word processors: problems and prospects. *ACM Trans. Inf. Syst*, *1*(3), 254-271.

Mackay, W. E. (1988). *More than just a communication system: diversity in the use of electronic mail*. CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work, New York, NY, USA, 344-353.

Mackay, W. E. (1990). *Patterns of sharing customizable software*. CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work, New York, NY, USA, 209-221.

Mackay, W. E. (1991). *Triggers and barriers to customizing software*. CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 153-160.

Mackay, W. E. & Fayard, A.-L. (1997). *HCI, natural science and design: a framework for triangulation across disciplines*. DIS '97: Proceedings of the conference on Designing interactive systems, New York, NY, USA, 223-234.

MacLean, A., Carter, K., Lövstr, L., & Moran, T. (1990). *User-tailorable systems: pressing the issues with buttons*. CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 175-182.

Macminute.com. (2004). GarageBand community Web sites ready to launch. Retrieved 23.5.2006, from http://www.macminute.com/2004/01/09/garageband

Maeda, J. (2005a). Less is Greater Than Zero. *Thoughts on Simplicity* Retrieved 28.7.2006, from http://weblogs.media.mit.edu/SIMPLICITY/archives/000230.html#law

Maeda, J. (2005b, September 7). Simplicity: The Goldilocks Rule. *BusinessWeek,*. McGraw Hill Companies Inc.

Maeda, J. (2005c). First Law of Simlicity. *Thoughts on Simplicity* Retrieved 28.7.2006, from Maeda http://weblogs.media.mit.edu/SIMPLICITY/archives/000113.html#firstlaw

Malone, T. W. (1983). How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. Inf. Syst*, *1*(1), 99-112.

Malone, T. W. (1984). Heuristics for designing enjoyable user interfaces: lessons from computer games. *Human factors in computer systems*. (pp. 1-12). Ablex Publishing Corp. Norwood, NJ, USA.

Marcus, A. (1995). Principles of effective visual communication for graphical user interface design. *Human-computer interaction: toward the year 2000*. (pp. 425-441). Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Martell, D. (2006, July 19). Apple profit rises 48 pct, helped by iPod sales. *Washington Post*.

Martin, D. & Sommerville, I. (2004). Patterns of cooperative interaction: Linking ethnomethodology and design. *ACM Trans. Comput.-Hum. Interact*, *11*(1), 59-89.

Mayhew, D. J. (1999). *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design (The Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann.

McCoy, K. (1995). Graphic Design in a Multicultural World. *HOW: The Bottomline Design Magazine*, *X*(2), 146-147.

McCrickard, D. S., Chewar, C. M., & Somervell, J. (2004). *Design, science, and engineering topics?: teaching HCI with a unified method*. SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education, New York, NY, USA, 31-35.

McDonald, J. E., Stone, J. D., Liebelt, L. S., & Karat, J. (1982). *Evaluating a method for structuring the user-system interface*. Proceedings the 26th Annual Meeting of the Human Factors Society.

McGrenere, J. (2002). *The Design and Evaluation of Multiple Interfaces: A Solution for Complex Software*. University of Toronto, Toronto. Available online at http://www.cs.ubc.ca/~joanna/papers/JMcGrenere_Thesis_DoubleSided.pdf

McGrenere, J., Baecker, R. M., & Booth, K. S. (2002). *An evaluation of a multiple interface design solution for bloated software*. CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 164-170.

McGrenere, J. & Moore, G. (2000). *Are We All In the Same "Bloat"?* Graphics Interface, 187-196. Available online at citeseer.ist.psu.edu/685361.html

Meij, H. v. d. (2003). Minimalism revisited. *Document Design*, *4*(3), 212-233.

Mencken, H. L. (1917, November 16). The Divine Afflatus. *New York Evening Mail*.

Mertens, M. & Leggewie, C. (2004, May 28). Technologisches Kokain. *Freitag - Die Ost-West-Wochenzeitung,*. Zeitungsverlag »Freitag« GmbH.

Mertens, W. (1983). *American Minimal Music*. London: Kahn & Averill.

Metafacts, I. (2003). Roadblocks on the Information Highway: Barriers to adoption of technology products.

Meyer, J. S. (2001). *Minimalism : art and polemics in the sixties*. New Haven: Yale University Press.

Meyer, T. & Münte-Goussar, S. (2005). study.log - angewandte Bildwissenschaft? *BDK-Mitteilungen*, *3*, 34-35.

Microsoft. (2006a). Powerful and Simple. Retrieved 13.10.2006, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/UxGuide/UXGuide/Principles/PowerfulAndSimple/PowerfulAndSimple.asp

Microsoft. (2006b). Windows Vista User Experience Guidelines. Retrieved 2.8.2006, from http://msdn.microsoft.com/library/?url=/library/en-us/UxGuide/UXGuide/Home.asp

Microsoft. (2006c). Microsoft Robotics Studio Provides Common Ground for Robotics Innovation. Retrieved 20.8.2006, from http://www.microsoft.com/presspass/press/2006/jun06/06-20MSRoboticsStudioPR.mspx

Microsoft. (1995). *The Windows Interface Guidelines for Software Design: An Application Design Guide*. Microsoft Press.

Microsoft. (1999). *Microsoft Windows User Experience (Microsoft Professional Editions)*. Microsoft Press.

Mijksenaar, P. & Westendorp, P. (1999). *Open Here: The Art of Instructional Design*. Joost Elffers Books.

Miles-Board, T., Carr, L., Kampa, S., & Hall, W. (2003). *Supporting management reporting: a writable web case study*. WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, 234-243.

Minard, C. J. (1869). Carte figurative des pertes successives en hommes de l'armée française dans la campagne de Russie. 1812-1813.: ENPC: Fol 10975, 10974/C612.

Mitchell, J. & Shneiderman, B. (1989). Dynamic versus static menus: an exploratory comparison. *SIGCHI Bull*, *20*(4), 33-37.

Mogensen, P., Bødker, S., & Grønbæk, K. (2000). Cooperative Design. In A. Kent (Ed.), *Encyclopedia of Library and Information Science vol 67*. (pp. 108-119). New York: Marcel Dekker.

Molich, R. & Nielsen, J. (1990). Improving a human-computer dialogue. *Commun. ACM*, 3), 338-348.

Monk, A., Hassenzahl, M., Blythe, M., & Reed, D. (2002). *Funology: designing enjoyment*. CHI '02: CHI '02 extended abstracts on Human factors in computing systems, New York, NY, USA, 924-925.

Moran, T. P. (1981). The command language grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, *15*(1), 3-50.

Morgan, C. (2005, March). GarageBand 2. *Computer Music*, pp. 88-89. Future Publishing Ltd.

Morkes, J. & Nielsen, J. (1998). Applying Writing Guidelines to Web Pages. Retrieved 22.4.2006, from http://www.useit.com/papers/webwriting/rewriting.html

Morris, R. (1966). Notes on Sculpture, part 2. *Artforum*, Reprinted in: Battcock, Gregory (ed.): Minimal art: a critical anthology. University of California Press, Berkeley, Los Angeles and London, 228-235, 1995.

Mossberg, W. S. (2001, June 7). New Windows XP Feature Can Re-Edit Others' Sites. *Wall Street Journal,*. Dow Jones & Company, Inc.

Müller-Prove, M. (2002). *Vision and Reality of Hypertext and Graphical User Interfaces*. Universität Hamburg, Bericht FBI-HH-B-237/02, Hamburg, Germany. Available online at http://www.mprove.de/diplom/

Mutius, B. v. (2004). *Die andere Intelligenz - Wie wir morgen denken werden*. Klett-Cotta.

Myers, B., Hollan, J., Cruz, I., Bryson, S., Bulterman, D., Catarci, T., Citrin, W., Glinert, E., Grudin, J., & Ioannidis, Y. (1996). Strategic directions in human-computer interaction. *ACM Comput. Surv*, *28*(4), 794-809.

Myers, B. A. (1998). A Brief History of Human Computer Interaction Technology. *ACM interactions*, *5*(2), 44-54.

Myers, B. A., Ko, A. J., & Burnett, M. M. (2006). *Invited research overview: end-user programming*. CHI '06: CHI '06 extended abstracts on Human factors in computing systems, New York, NY, USA, 5-7.

Myers, B. A., Weitzman, D. A., Ko, A. J., & Chau, D. H. (2006). *Answering why and why not questions in user interfaces*. CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA, 397-406.

Mylopoulos, J., Chung, L., & Yu, E. (1999). From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, *42*(1), 31-37.

Nagel, D. (2005). Apple GarageBand 2: Home studio audio and MIDI software. Retrieved 22.5.2006, from http://www.macaudiopro.com/articles/viewarticle.jsp?id=31007

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. (2002). *Evolution patterns of open-source software systems and communities*. IWPSE '02: Proceedings of the International Workshop on Principles of Software Evolution, New York, NY, USA, 76-85.

Nanno, T., Saito, S., & Okumura, M. (2002). *Zero-Click : a system to support Web browsing*. International World Wide Web Conference, Honululu, Hawaii, 90-91.

Nardi, B. A. (1993). *A Small Matter of Programming: Perspectives on End User Computing*. The MIT Press.

Nardi, B. A. & Miller, J. R. (1990a). *An ethnographic study of distributed problem solving in spreadsheet development*. CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work, New York, NY, USA, 197-208.

Nardi, B. A. & Miller, J. R. (1990b). *The spreadsheet interface: A basis for end user programming*. INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction, 977-983.

Naur, P. (1985). Programming As Theory Building. In P. Naur (Ed.), *Computing: A Human Activity*. (pp. 37-48). Reading: Addison Wesley.

Ltd., N. (2005). FDD overview.

Nelson, T. H. (1990). The Right Way to Think About Software Design. In B. Laurel & D. Mills (Eds.), *The Art of Human-Computer Interface Design*. Ontario: Addison-Wesley Publishing Company Inc.

Newman, B., O'Neill, J. P., & McNickle, M. (1990). *Barnett Newman : selected writings and interviews* (1st ed ed.). New York: Knopf : Distributed by Random House.

Nielsen, J. (1989). Coordinating user interfaces for consistency. *SICCHI Bulletin*, *20*, 63-65.

Nielsen, J. (1993). *Usability engineering*. Boston: Academic Press.

Nielsen, J. (2000). *Designing Web usability*. Indianapolis, IN: New Riders.

Nielsen, J. (1997). Be Succinct! (Writing for the Web). *Alertbox* Retrieved 20.4.2006, from http://www.useit.com/alertbox/9703b.html

Nielsen, J. (1999). Ten Good Deeds in Web Design. *Alertbox* Retrieved 21.4.2006, from http://www.useit.com/alertbox/991003.html

Nielsen, J. (2005). Forms vs. Applications. *Alertbox* Retrieved 21.4.2006, from http://www.useit.com/alertbox/forms.html

Nielsen, J. & Mack, R. L. (1994). *Usability inspection methods*. New York: Wiley.

Nielsen, J. & Tahir, M. (2002). *Homepage usability : 50 websites deconstructed*. Indianapolis, IN: New Riders.

Noirhomme-Fraiture, M. & Serpe, V. (1998). *Visual representation of hypermedia links according to their types*. AVI '98: Proceedings of the working conference on Advanced visual interfaces, New York, NY, USA, 146-155.

Nokia. (2002). Vertu Launches the World's Most Exclusive Instrument for Personal Communication. Retrieved 1.10.2006, from http://press.nokia.com/PR/200201/845684_5.html

Norman, D. A. (2004a). Activity-Centered Design: Why I like my Harmony Remote Control. Retrieved 13.8.2006, from http://jnd.org/dn.mss/activitycentere.html

Norman, D. A. (2004b). The truth about Google's so-called "simplicity". Retrieved 8.10.2006, from http://www.jnd.org/dn.mss/the_truth_about.html

Norman, D. A. (1981). Categorisation of action slips. *Psychological Review*, *88*(1), 1-15.

Norman, D. A. (1983). *Design principles for human-computer interfaces*. CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems, New York, NY, USA, 1-10.

Norman, D. A. (1999a). *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. The MIT Press.

Norman, D. A. (2002a). Emotion & design: attractive things work better. *interactions*, *9*(4), 36-42.

Norman, D. A. (2005). Do companies fail because their technology is unusable? *interactions*, *12*(4), 69.

Norman, D. A. (1989). *The Psychology of Everyday Things*. Basic Books.

Norman, D. A. (2004c). *Emotional Design: Why We Love (Or Hate) Everyday Things*. Basic Books.

Norman, D. A. & Draper, S. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates.

Norman, D. A. (1999b). Affordance, conventions, and design. *interactions*, *6*(3), 38-43.

Norman, D. A. (2002b). Home theater: Not ready for prime time. *Computer*, *35*(6), 100-102.

Norman, K. L. (1991). *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface (Human/Computer Interaction)*. Ablex Publishing Corporation.

Nozick, R. (1974). *Anarchy, state, and utopia*. New York: Basic Books.

Nuseibeh, B. & Easterbrook, S. (2000). *Requirements engineering: a roadmap*. ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, 35-46.

Nyman, M. (1972). S R-mysteries of the phase. *Music and Musicians, June 1972*, 20-21.

Nyman, M. (1974). *Experimental Music: Cage and Beyond*. New York: Schirmer.

O'Bryan, L. (2002). *Minimalist Interiors*. Atrium Publishers Group.

O'Day, V. L., Bobrow, D. G., Hughes, B., Bobrow, K. B., Saraswat, V. A., Talazus, J., Walters, J., & Welbes, C. (1996). *Community designers*. Proc. PDC'96, 3-13.

O'Reilly, T. (1999). Lessons from open-source software development. *Commun. ACM*, *42*(4), 32-37.

O'Reilly, T. (2005). What is Web 2.0? Design Patterns and Business Models for the Next Generation of Software. Retrieved 10.10.2006, from http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Obendorf, H., Schmolitzky, A., & Finck, M. (2006). *XPnUE – defining and teaching a fusion of eXtreme programming and usability engineering*. HCI Educators Workshop 2006: HCIEd.2006-1 inventivity: Teaching theory, design and innovation in HCI, Limerick, Ireland,. Available online at http://www.idc.ul.ie/hcieducators06/Procs

Obendorf, H. (2003). *Einsatz elektronischer Medien zur Unterstützung der universitären Lehre: Ein Erfahrungsbericht*. Mensch & Computer 2003, Stuttgart, 356-357.

Obendorf, H. (2004). The Indirect Authoring Paradigm - Bringing Hypertext into the Web. *J. Digit. Inf, 5*(1).

Obendorf, H. (2005). *A Minimal Design Game*. International Forum 'Less is more - Simple Computing in an Age of Complexity', Cambridge, UK, 1-10. Available online at http://research.microsoft.com/conferences/LessIsMore/

Obendorf, H., Finck, M., Janneck, M., & Bødker, S. (in prep.). Using PD for Intercontextual Community Building: Communicating Design Philosophy and Enriching the User Experience. *Prepared for Scandinavian Journal on Information Systems (SJIS)*.

Obendorf, H., Finck, M., & Janneck, M. (in prep.). What Shall We Link? – Use-Centered Design for Community Support: A Case Study. *Prepared for Journal for Online Digital Information (JoDI)*.

Obendorf, H., Finck, M., & Schmolitzky, A. (2005). Teaching balance and respect: HCI Group \& Software Technology Group at the University of Hamburg. *interactions*, *12*(5), 36-37.

Obendorf, H. & Weinreich, H. (2003). *Comparing Link Marker Visualization Techniques – Changes in Reading Behavior*. Proc. of 12th International World Wide Web Conference - WWW 2003, New York, NY, USA, 736-745.

Obendorf, H., Weinreich, H., Herder, E., & Mayer, M. (submitted). Web Page Revisitation Revisited: Implications of a Long-term Click-stream Study of Web Browser Usage. *Submitted to CHI 2007*.

Oberquelle, H. (2005). *Gestaltung von Benutzungsschnittstellen: Geht es ohne Design-Kompetenz?* INFORMATIK 2005 - Informatik LIVE! Band 1, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Bonn, 19. bis 22. September 2005, 232-237.

Olsen, H. (2003). Balancing visual and structural complexity in interaction design: How visual simplicity can harm usability. Retrieved 10.10.2006, from http://www.guuui.com/issues/04_03.php

Orlowski, A. (2004, 6.1.). Jobs caps snoozathon with cut-down Emagic, iPod. *The Register*.

Orlowski, A. (2005). Web 2.0: It's. like your brain on LSD!. Retrieved 28.8.2006, from http://www.theregister.co.uk/2005/10/21/web_two_point_nought_poll/

Osborne, H. (1988). *The Oxford companion to twentieth-century art*. Oxford [Oxfordshire] ; New York: Oxford University Press.

Ossenbrügge, K. (2004a). Das Handy, ein Mysterium für grey consumers? *Research International*.

Ossenbrügge, K. (2004b). Kaufblockaden und wie sie zu therapieren sind. *Research International*.

Page, S. R., Johnsgard, T. J., Albert, U., & Allen, C. D. (1996). *User customization of a word processor*. CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 340-346.

Palmer, S. & Felsing, M. (2002). *A Practical Guide to Feature Driven Development*. Prentice Hall PTR.

Pape, B., Bleek, W.-G., Jackewitz, I., & Janneck, M. (2002). *Software Requirements for Project-Based Learning - CommSy as an Exemplary Approach*. Proc. HICSS 2002.

Pape, B., Janneck, M., & Klein, M. (2005). Matching Software and Context in Open Learning Scenarios – The Use of Log File Analysis to Evaluate Didactical Fit. *e-learning and education (eleed)*, 1).

Pape, B. (2005). *Organisation der Softwarenutzung - Theoriebildung und Fallstudien zu Softwareeinführung und Benutzungsbetreuung*. Berlin: Logos.

Pape, B., Krause, D., & Oberquelle, H. (2004). *Wissensprojekte : gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*. Münster [u.a.]: Waxmann.

Parker, I. (2001). Absolute PowerPoint. *The New Yorker*, pp. 76-87.

Parks, M. R. & Floyd, K. (1996). Making Friends in Cyberspace. *Journal of Computer Mediated Communication*, *1*(4).

Patton, M. Q. (2001). *Qualitative Research & Evaluation Methods*. Sage Publications, Inc.

Paul, M. & Rob, B. (2000). Intermediaries personalize information streams. *Commun. ACM*, *43*(8), 96-101.

Perez, J. C. (2005). Q&A: Vint Cerf on Google's challenges, aspirations. Retrieved 20.8.2006, from http://www.computerworld.com/developmenttopics/development/story/0,10801,106535,00.html

Perin, C. (1991). Electronic social fields in bureaucracies. *Commun. ACM*, *34*(12), 75-82.

Perreault, J. (1968). Simple, Not Simple-Minded. *Village Voice*, 16-17.

Perreault, J. (1967). The Term Minimal. *Arts*, Reprinted in: Battcock, Gregory (ed.): Minimal art: a critical anthology. University of California Press, Berkeley, Los Angeles and London, 260-261, 1995.

Petersen, M. G., Iversen, O. S., Krogh, P. G., & Ludvigsen, M. (2004). *Aesthetic interaction: a pragmatist's aesthetics of interactive systems*. DIS '04: Proceedings of the 2004 conference on Designing interactive systems, New York, NY, USA, 269-276.

Peterson, Z. & Burns, R. (2005). Ext3cow: a time-shifting file system for regulatory compliance. *Trans. Storage*, *1*(2), 190-212.

Petterson, E. (2004). *New Minimalist Architecture*. Collins Design.

Petterson, E. (2003). *Minimalist Architecture*. Atrium Publishers Group.

phpBB. (2006). phpBB.com: Creating Communities. Retrieved 17.7.2006, from http://www.phpbb.com/

Pickering, J. M. & King, J. L. (1992). *Hardwiring weak ties: individual and institutional issues in computer mediated communication*. CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work, New York, NY, USA, 356-361.

Pipek, V. (2005). *From Tailoring to Appropriation Support: Negotiating Groupware Usage*. University of Oulu, Oulu, Finland. Available online at http://herkules.oulu.fi/isbn9514276302/

Poe, E. A. (1842, April). Twice-Told Tales. Graham's Magazine, Reprinted in "Edgar Allan Poe: Essays and Reviews", The Library of America, 1984, 568-569.

Pogue, D. (2004). *GarageBand: The Missing Manual (Missing Manuals)*. Pogue Press.

Pogue, D. & Story, D. (2006). *iPhoto 6: The Missing Manual (Missing Manual)*. O'Reilly Media.

Pohl, K. (1997). Requirements Engineering: An Overview. In A. Kent & J. Williams (Eds.), *Encyclopedia of Computer Science and Technology*. (pp. Volume 36, Supplement 21). New York: Marcel Dekker, Inc.

Pope, F. (1888, April). The American Inventors of the Telegraph, with special references to the services of Alfred Vail. *The Century: Illustrated Monthly Magazine*.

Potter, K. (2000). *Four Musical Minimalists: La Monte Young, Terry Riley, Steve Reich, Philip Glass (Music in the Twentieth Century S.)*. Cambridge University Press.

Potter, K. & Smith, D. (1976). Interview with Philip Glass. *Contact*, *13*, 25.

Preece, J. (1994). *Human Computer Interaction (ICS S.)*. Addison Wesley.

Press, L. (1993). Before the Altair: the history of personal computing. *Commun. ACM*, *36*(9), 27-33.

Preve, F. (2004). *Power Tools for GarageBand: Creating Music with Audio Recording, MIDI Sequencing, and Loops*. Backbeat Books.

Quint, V. & Vatton, I. (2005). *Towards active web clients*. DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering, New York, NY, USA, 168-176.

Ragtime. (2005). Etwas Einzigartiges schaffen. Retrieved 5.10.2006, from http://www.ragtime.de/Content/firmengeschichte/index.html

Ragtime. (2006a). Die RagTime GmbH stellt RagTime Privat ein. Retrieved 8.10.2006, from http://www.ragtime.de/link.cgi?presse_mitteilung&anc=rt6privat&pg=presse_mitteilung.html

Ragtime. (2006b). Neu in Ragtime 6.

Raskin, J. (1982). Computers by the Millions. *SIGPC Newsletter*, *5*(2).

Raskin, J. (2000a). The Humane Interface (book excerpt). *Ubiquity*, *1*(14), 3.

Raskin, J. (2000b). *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering*. Amsterdam, The Netherlands: North-Holland.:.

Reason, J. a. M., Klara. (1982). *Absent Minded?: The Psychology of Mental Lapses, Little Slips, and Everyday Errors*. Prentice Hall.

Reason, R. (2001). Typography: Less Is More. Retrieved 12.10.2006, from http://www.poynter.org/dg.lts/id.33/aid.3308/column.htm

Reeder, R. W. & Maxion, R. A. (2005). *User Interface Dependability through Goal-Error Prevention*. DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05), 60-69. Available online at http://www.cs.cmu.edu/afs/cs.cmu.edu/user/maxion/www/pubs/ReederMaxionDSN05.pdf

Reinhardt, A. (1975). *Art-as-Art: Selected Writings of Ad Reinhardt*. New York: Viking.

Reisner, P. (1990). *What is inconsistency?* INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction, 175-181.

Rettig, M. (1994). Prototyping for tiny fingers. *Commun. ACM*, *37*(4), 21-27.

Rickman, J. T., Sunkel, M. J., & Hobbs, J. (1979). *Word processing and data entry computing services*. SIGUCCS '79: Proceedings of the 7th annual ACM SIGUCCS conference on User services, New York, NY, USA, 68-72.

Rieman, J., Lewis, C., Young, R. M., & Polson, P. G. (1994). *Why is a raven like a writing desk?: lessons in interface consistency and analogical reasoning from two cognitive architectures*. CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 438-444.

Riley, T. (1964). In C.

Robbins, A. (1999). *Unix in a Nutshell: A Desktop Quick Reference for SVR4 and Solaris 7 (3rd Edition)*. O'Reilly Media, Inc.

Robbins, A. (2005). *Unix in a Nutshell, Fourth Edition*. O'Reilly Media, Inc.

Robbins, A. & Beebe, N. H. F. (2005). *Classic Shell Scripting*. O'Reilly Media, Inc.

Rockwell, J. (1986, December 21). The Death and Life of Minimalism. *New York Times*, p. 29.

Rohall, S. L., Gruen, D., Moody, P., Wattenberg, M., Stern, M., Kerr, B., Stachel, B., Dave, K., Armes, R., & Wilcox, E. (2004). *ReMail: a reinvented email prototype*. CHI '04: CHI '04 extended abstracts on Human factors in computing systems, New York, NY, USA, 791-792.

Rolfe, R. (2003). What is an IME (Input Method Editor) and how do I use it? Retrieved 20.7.2006, from http://www.microsoft.com/globaldev/handson/user/IME_Paper.mspx

Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N. A. M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., & Heymans, P. (1998). A Proposal for a Scenario Classification Framework. *Requirements Engineering Journal, 3*(1), 23-47.

Rose, B. (1987). Interview with Robert Rauschenberg. *Rauschenberg*. New York: Vintage.

Rose, B. (1965a). Looking at American Sculpture. *Artforum*, 29-36.

Rose, B. (1965b). ABC Art. *Art in America*, 57-69.

Rosenberg, H. (1964). *The Anxious Object*. Chicago: University of Chicago Press.

Rosenblum, R. (1965). Frank Stella. *Artforum, III*(6), 20-25.

Rosenfeld, L. & Morville, P. (2002). *Information Architecture for the World Wide Web*. O'Reilly.

Rosenfeld, L. & Morville, P. (1998). *Information architecture for the World Wide Web* (1st ed ed.). Cambridge ; Sebastopol, CA: O'Reilly.

Rosenthal, H. (2006). *Discovering Automator*. BookSurge Publishing.

Rossell, Q. (2005). *Minimalist Interiors*. Collins Design.

Rosson, M. B. (1999). Integrating development of task and object models. *Commun. ACM, 42*(1), 49-56.

Rosson, M. B. & Carroll, J. M. (2001). *Usability Engineering Scenario-based Development of Human Computer Interaction*. Morgan Kaufmann Publishers Inc,US.

Rosson, M. B., Carroll, J. M., & Rodi, C. M. (2004). *Case studies for teaching usability engineering*. SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education, New York, NY, USA, 36-40.

Rothstein, E. (1995). *Emblems of mind : the inner life of music and mathematics* (1st ed ed.). New York: Times Books/Random House.

Rozanski, E. P. & Haake, A. R. (2003). *The many facets of HCI*. CITC4 '03: Proceedings of the 4th conference on Information technology curriculum, New York, NY, USA, 180-185.

RPO (2005, 23.12.). Moderne Technik überfordert häufig. Rheinische Post Online.

Board, R. S. S. A. (2006). Really Simple Syndication. Retrieved 10.6.2006, from http://www.rssboard.org/rss-specification

Rubin, C. (1999). *Running Microsoft Word 2000 (Running)*. Microsoft Press.

Sartre, J.-P. (1965). *What is Literature?*. New York: Harper.

Scaife, M., Rogers, Y., Aldrich, F., & Davies, M. (1997). *Designing for or designing with? Informant design for interactive learning environments*. CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 343-350.

Schulze, F. (1986). *Mies Van Der Rohe*. Germany: Ernst,Wilhelm & Sohn,Verlag fur Architektur und Technische Wissenschaften Gmbh.

Schumacher-Rasmussen, E. (2004). Review: Apple GarageBand. Retrieved 23.5.2006, from http://www.eventdv.net/Articles/ReadArticle.aspx?CategoryID=52&ArticleID=8707

Schwaber, K. (2004). *Agile Project Management with Scrum (Microsoft Professional)*. Microsoft Press.

Schwaber, K. & Beedle, M. (2001). *Agile Software Development with SCRUM*. Prentice Hall.

Schwartz, J. (2003, 28 September). The level of discourse continues to slide. *The New York times*, p. 12 col. 1.

Schwartz, K. R. (1996). *Minimalists (20th-century Composers S.)*. Phaidon Press.

Sears, A. & Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. *ACM Trans. Comput.-Hum. Interact*, *1*(1), 27-51.

Post-Intelligencer, S. (2006). Google admits online stumble. Retrieved 13.10.2006, from http://seattlepi.nwsource.com/business/257040_googlestore26.html

Secrist, L. J. (1980). *Should we get involved in word processing?* SIGUCCS '80: Proceedings of the 8th annual ACM SIGUCCS conference on User services, New York, NY, USA, 60-61.

Seffah, A. & Metzker, E. (2004). The obstacles and myths of usability and software engineering. *Commun. ACM*, *47*(12), 71-76.

Sellers, D. (2005). 'Macsimum News' interviews creator of iPhoto, iMovie, Bubbler. Retrieved 25.5.2006, from http://www.macsimumnews.com/index.php/archive/macsimum_news_interviews_creator_of_iphoto_imovie_bubbler/

Sengers, P. & Gaver, B. (2006). *Staying open to interpretation: engaging multiple meanings in design and evaluation*. DIS '06: Proceedings of the 6th ACM conference on Designing Interactive systems, New York, NY, USA, 99-108.

SEP. (2003). Feminist Epistemology and Philosophy of Science: Feminist Standpoint Theory. *Stanford Encyclopedia of Psychology* Retrieved 13.10.2006, from http://plato.stanford.edu/entries/feminism-epistemology/#standpoint

Sergio, F. (2001). FreeGorifero. Retrieved 10.10.2006, from http://www.freegorifero.com/weblog/2002_11_01_weblog_archive.html

Serra, R. & Weyergraf-Serra, C. (1980). *Richard Serra, interviews, etc., 1970-1980*. Yonkers, N.Y. (Trevor Park-on-Hudson) [S.l.]: Hudson River Museum Distributed by Art Catalogues.

Seuren, P. A. M. (2004). *Chomsky's minimalism*. New York: Oxford University Press.

Sharon, T., Lieberman, H., & Selker, T. (2003). *A zero-input interface for leveraging group experience in web browsing*. IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces, New York, NY, USA, 290-292.

Shaw, R. (2005). Web 2.0? It doesn't exist. Retrieved 9.10.2006, from http://blogs.zdnet.com/ip-telephony/?p=805

Shedroff, N. (2001). *Experience Design 1*. Waite Group Press.

Shneiderman, B. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Reading, Mass: Addison-Wesley.

Shneiderman, B. (1997). *Direct manipulation for comprehensible, predictable and controllable user interfaces*. IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces, New York, NY, USA, 33-39.

Shneiderman, B. (2003). *Promoting universal usability with multi-layer interface design*. CUU '03: Proceedings of the 2003 conference on Universal usability, New York, NY, USA, 1-8.

Shneiderman, B. & Plaisant, C. (2004). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (4th edition). Addison Wesley.

Shusterman, R. (1999). Somaesthetics: A Disciplinary Proposal. *Journal of Aesthetics and Art Criticism*, *57*.

Sievers, D. (2004). Say NO by default. Retrieved 17.7.2006, from \http://www.oreillynet.com/pub/wlg/5384

Simons, T. (2004). Does PowerPoint make you stupid? *Presentations Magazine*, pp. 24-31. VNU Business Media.

Siracusa, J. (2001). Macworld Expo San Francisco 2001. Retrieved 21.7.2006, from http://arstechnica.com/reviews/01q1/macwldsf/mwsf-7.html#itunes

Sixtus, M. (2006, 10). Jenseits von gut und böse: Der unheimliche Erfolg von Google. *c't Magazin für Computertechnik*, pp. 162-167. heise.

SketchUp. (2006). SketchUpdate 04.27.06. http://de.sketchup.com/index.php?id=1444

Sloan, M. E. (1975). *A design-oriented computer engineering program*. ISCA '75: Proceedings of the 2nd annual symposium on Computer architecture, New York, NY, USA, 220-224.

Smith, D. (1977). Following a Straight Line: La Monte Young. *Contact*, *18*, 4-9.

Smith, D. C., Irby, C., Kimball, R., & Verplank, B. (1982). Designing the Star User Interface. *Byte*, *4*, 242-282.

Smith, G. & Smith, N. W. (1994). *American originals : interviews with 25 contemporary composers*. London ; Boston: Faber and Faber.

Smith, S. L. & Mosier, J. N. (1986). Guidelines for designing user interface software. *Report ESD-TR-86-278*.

Smith, T. (2005). Apple sued over iTunes - again. *The Register* Retrieved 13.10.2006, from http://www.theregister.co.uk/2005/06/21/apple_sued_over_itunes/

Snyder, C. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces (The Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann.

Souza, F. d. & Bevan, N. (1990). *The use of guidelines in menu interface design: Evaluation of a draft standard*. INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction, 435-440.

Spector, N. (1974). Essential Painting. *Robert Mangold*. La Jolla: La Jolla Museum of Contemporary Art.

Steinberg. (2006a). Cubase SL 3. Retrieved 2.10.2006, from http://www.steinberg.de/34_1.html

Steinberg. (2006b). Feature Comparison. Retrieved 2.10.2006, from http://www.steinberg.net/552_1.html

Stephens, M. (2005). The iPod Experiments. *LibraryJournal.com* http://www.libraryjournal.com/article/CA515808.html

Stone, M. & Foss, B. (2001). *Successful Customer Relationship Marketing*. Kogan Page Business Books.

Straus, D. & Doyle, M. (1978). The Architect as Facilitator: A New Role. *Journal of Architectural Education (JAE)*, *31*(4), 13-15.

Strauss, M., Pape, B., Adam, F., Klein, M., & Reinecke, L. (2003). CommSy-Evaluationsbericht 2003: Softwareunterstützung für selbstständiges und kooperatives Lernen. *Berichte des Fachbereichs Informatik der Universität Hamburg, FBI-HH-B-251/03*.

Strickland, E. (1991). *American Composers: Dialogues on Contemporary Music*. Bloomington: Indiana University Press.

Strickland, E. (2000). *Minimalism: Origins* (2nd edition ed.). Bloomington: Indiana University Press.

Suchman, L. (1995). Making work visible. *Commun. ACM*, *38*(9), 56-ff.

Suchman, L. A., Pea, R., Brown, J. S., & Heath, C. (1987). *Plans and Situated Actions : The Problem of Human-Machine Communication (Learning in Doing: Social, Cognitive & Computational Perspectives)*. Cambridge University Press.

Sutherland, I. E. (1963). *Sketchpad: A man-machine graphical communication system*. MIT,. Available online at http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-574.pdf

Tassabehji, R. & Vakola, M. (2005). Business email: the killer impact. *Commun. ACM*, *48*(11), 64-70.

Terdiman, D. (2005). IPod Shuffle Sparks Stampede. *Wired Magazine*.

Tessler, F. (2006). Keynote 3: Presentation software keeps improving with age. *Macworld* Retrieved 17.7.2006, from http://www.macworld.com/2006/01/reviews/keynote3/index.php

Tetzlaff, L. & Schwartz, D. R. (1991). *The use of guidelines in interface design*. CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 329-333.

Thompson, D. V., Hamilton, R. W., & Rust, R. T. (2005). Feature Fatigue: When Product Capabilities Become Too Much of a Good Thing. *Journal of Marketing Research*, *42*, 431-442.

Tischler, L. & Mayer, M. (2005). The Beauty of Simplicity. *Fast Company*, *100*, 52.

Tognazzini, B. (1989). Achieving consistency for the Macintosh. In J. Nielsen (Ed.), *Coordinating user interfaces for consistency*. (pp. 57-73). San Diego, CA: Academic Press Professional, Inc.

Tognazzini, B. (1993). *Principles, techniques, and ethics of stage magic and their application to human interface design*. CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 355-362.

Tognazzini, B. (1992). *Tog on Interface*. Addison-Wesley Professional.

Tomkins, C. (1981). *Off the Wall: Robert Rauschenberg and the Art World of Our Time*. New York: Penguin Books.

Tractinsky, N. (1997). *Aesthetics and apparent usability: empirically assessing cultural and methodological issues*. CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 115-122.

Trigg, R. H. & Bødker, S. (1994). *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work, New York, NY, USA, 45-54. Available online at citeseer.ist.psu.edu/trigg94from.html

Trout, J. & Rivkin, S. (1998). *The Power of Simplicity*. McGraw-Hill Companies.

Trussler, M. (1994). The narrowed voice: Minimalism and Raymond Carver. *Studies in Short Fiction, 31*.

Tsandilas, T. & schraefel, m. c. (2005). *An empirical assessment of adaptation techniques*. CHI '05: CHI '05 extended abstracts on Human factors in computing systems, New York, NY, USA, 2009-2012.

Tschichold, J. (1925). Elementare Typografie. *Typographische Mitteilungen, Oktoberheft/Sonderheft.*

Tschichold, J. (1928). *Die neue Typographie : ein Handbuch für zeitgemäss Schaffende* (1. bis 5. Tsd ed.). Berlin: Verl. des Bildungsverb. der Dt. Buchdrucker.

Tufte, E. (2003). PowerPoint Is Evil: Power Corrupts. PowerPoint Corrupts Absolutely. *Wired Magazine, 11.09*.

Tufte, E. R. (1986). *Visual Display of Quantitative Information*. Graphics Press.

Tufte, E. R. (1990). *Envisioning Information*. Graphics Press.

Tufte, E. R. (2006). *The Cognitive Style of PowerPoint: Pitching Out Corrupts Within Second Edition*.

Tulga, P. & Tulga, D. (2005). Morse Code Music - connecting rhythm and language with Morse Code. Retrieved 3.8.2006, from http://www.philtulga.com/morse.html

Tullis, T. S. (1988). Screen design. In M. Helander (Ed.), *Handbook of Human-Computer Interaction*. (pp. 377--411).

Uren, V., Shum, S. B., Li, G., Domingue, J., & Motta, E. (2003). *Scholarly publishing and argument in hyperspace*. WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, 244-250.

Varian, E. (1967). Schemata 7 catalogue.

Venturi, R., Scott, B., Denise, & Izenour, S. (1972). *Learning from Las Vegas*. Cambridge, Mass: MIT Press.

Vidgen, R. & Braa, K. (1997). Balancing Interpretation and Intervention in Information System Research: The Action Case Approach. In A. S. Lee, J. Liebenau, & J. I. DeGross (Eds.), *Information Systems and Qualitative Research,*. (pp. 524 - 541). London:.

Virzi, R. A., Sokolov, J. L., & Karis, D. (1996). *Usability problem identification using both low- and high-fidelity prototypes*. CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 236-243.

Voelcker, J. (1986). The Information Appliance. *IEEE Spectrum*, *23*(5), 65.

Voida, A., Grinter, R. E., Ducheneaut, N., Edwards, W. K., & Newman, M. W. (2005). *Listening in: practices surrounding iTunes music sharing*. CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 191-200.

Vredenburg, K., Mao, J.-Y., Smith, P. W., & Carey, T. (2002). *A survey of user-centered design practice*. CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 471-478.

Wales, J. (2005). *Wikipedia in the free culture revolution*. OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, NY, USA, 5-5.

Wanner, A. (2003). *Russian minimalism : from the prose poem to the anti-story* (Studies in Russian literature and theory). Evanston, Ill: Northwestern University Press.

Weinberger, D. (2002). *Small pieces, loosely joined : a unified theory of the web*. Cambridge, MA: Perseus Pub.

Weinreich, H., Buchmann, V., & Lamersdorf, W. (2003). *Scone: Ein Framework zur evaluativen Realisierung von Erweiterungen des Webs*. Tagungsband Kommunikation in Verteilten Systemen - KiVS 2003, 31-42.

Weinreich, H., Obendorf, H., Herder, E., & Mayer, M. (2006a). *Der Wandel in der Benutzung des World Wide Web*. Mensch & Computer 2006.

Weinreich, H., Obendorf, H., Herder, E., & Mayer, M. (2006b). *Off the beaten tracks: exploring three aspects of web navigation*. WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, 133-142.

Weinreich, H., Obendorf, H., Herder, E., & Mayer, M. (to appear). Not Quite the Average: An Empirical Study of Web Use. *ACM Transactions on the Web*.

Weinreich, H., Obendorf, H., & Lamersdorf, W. (2004). HyperScout: Linkvorschau im World Wide Web. *i-com: Zeitschrift für interaktive und kooperative Medien*, *1/2004*(3), 4-12.

Weinreich, H., Obendorf, H., & Lamersdorf, W. (2001). *The Look of the Link - Concepts for the User Interface of Extended Hyperlinks*. Proceedings of the 12th ACM Conference on Hypertext and Hypermedia, University of Aarhus, Århus, Denmark, New York, NY, USA, 19-28.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, *36*(7), 75-84.

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, 94-104.

Weiser, M. & Brown, J. S. (1997). The coming age of calm technolgy. *Beyond calculation: the next fifty years*. (pp. 75-85). Copernicus New York, NY, USA.

Welzer, H. (2006). Nur nicht über Sinn reden!. *Die Zeit*, *40/2006*.

Wenger, E. (1999). *Communities of Practice: Learning, Meaning, and Identity* (*Learning in Doing: Social, Cognitive & Computational Perspectives S.)*. Cambridge University Press.

Wenger, E., McDermott, R. A., & Snyder, W. (2002). *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Harvard Business School Press.

Wenger, E. C. & Snyder, W. M. (2000). Communities of Practice: The Organizational Frontier. *Harvard Business Review*, *78*(1), 139-145.

Whittaker, S., Bellotti, V., & Gwizdka, J. (2006). Email in personal information management. *Commun. ACM*, *49*(1), 68-73.

Whittaker, S., Jones, Q., Nardi, B., Creech, M., Terveen, L., Isaacs, E., & Hainsworth, J. (2004). ContactMap: Organizing communication in a social desktop. *ACM Trans. Comput.-Hum. Interact*, *11*(4), 445-471.

Whittaker, S. & Sidner, C. (1996). *Email overload: exploring personal information management of email*. CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 276-283.

Wiecha, C. (1992). ITS and user interface consistency: a response to Grudin. *ACM Trans. Inf. Syst*, *10*(1), 112-114.

Wiecha, C., Bennett, W., Boies, S., & Gould, J. (1989). Tools for Generating Consistent User Interfaces. In J. Nielsen (Ed.), *Coordinating user interfaces for consistency*. (pp. 107-130). Boston, MA: Academic Press.

Wilchins, D. (1998, April 17). Getting inside design. *Yake Herald*.

Wilkinson, S. (2002). uDrive Me Crazy. Retrieved 10.10.2006, from http://www.popsci.com/popsci/printerfriendly/automotivetech/e0633bcc2eb84010vgnvcm1000004eecbccdrcrd.html

Williams, M. (2005). How Much Should an IPod Shuffle Cost? *PC World*.

Wilson, E. V. (2002). Email winners and losers. *Commun. ACM*, *45*(10), 121-126.

Winograd, T. & Flores, F. (1987). *Understanding Computers and Cognition : A New Foundation for Design*. Addison-Wesley Professional.

Winter, D. (1999). PONG-Story. Retrieved 10.10.2006, from http://www.pong-story.com/

Witt, R. J. & Tyerman, S. P. (2002). *Reducing cognitive overhead on the world wide web*. CRPITS '02: Proceedings of the twenty-fifth Australasian conference on Computer science, Darlinghurst, Australia, 311-320.

Wolf, T. V., Rode, J. A., Sussman, J., & Kellogg, W. A. (2006). *Dispelling "design" as the black art of CHI*. CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, New York, NY, USA, 521-530.

Wolfhagen, J. (2006). *Evaluation webbasierter Groupware im Kontext universitärer Lehre am Beispiel von BSCW und CommSy*. Unpublished Diplomarbeit, University of Hamburg, Hamburg.

Wollenweber, F. (2004). *Kollaborative Nutzung des World-Wide-Web*. Unpublished Diploma Thesis, University of Hamburg, Hamburg. Available online at http://vsys-www.informatik.uni-hamburg.de/getDoc.php/thesis/132/FrankWdiplomarbeit-final-070104.pdf

Wollheim, R. I. (1965). Minimal Art. *Arts Magazine*, *January 1965*, 26-32.

Wroblewski, D. A. (1991). The construction of human-computer interfaces considered as a craft. In J. Karat (Ed.), *Taking software design seriously*. (pp. 1-19). New York: Academic Press Professional, Inc. San Diego, CA, USA.

Wroblewski, L. (2005). 9 Lessons from 9 Years of Interface Design. Retrieved 22.6.2006, from http://www.lukew.com/ff/entry.asp?209

Wurman, R. S. (1991). *Information Anxiety*. Pan.

Wurman, R. S. & Bradford, P. (1996). *Information Architects*. Graphis Inc.

Xerox. (1983). Xerox Star and the Professional : Promotional video for the Xerox Star.

Young, L. M. (2001). Notes on Composition 1960 #7. Retrieved 13.10,.2006, from http://www.diapasongallery.org/archive/01_06_20.html

Young, L. M. & Zazeela, M. (1969). *Selected Writings*. Munich: Heiner Friefrich.

Zanino, M. C., Agarwal, R., & Prasad, J. (1994). *Graphical user interfaces and ease of use: some myths examined*. SIGCPR '94: Proceedings of the 1994 computer personnel research conference on Reinventing IS : managing information technology in changing organizations, New York, NY, USA, 142-154.

Zelevansky, L. (1991). Ad Reinhardt and the Younger Artists of the 1960s. In J. Elderfield (Ed.), *American Art of the 1960s*. New York: Museum of Modern Art.

Zeleznik, R. C., Herndon, K. P., & Hughes, J. F. (1996). *SKETCH: an interface for sketching 3D scenes*. SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, New York, NY, USA, 163-170.

Zevi, B. (1957). *Architecture As Space*. Horizon Press.

Zhou, Y. (2003). *The historical evolution of Chinese languages and scripts [Zhongguo yu wen de shi dai yan jin]* (Pathways to advanced skills ; vol. 8). Columbus, Ohio: National East Asian Languages Resource Center, Ohio State University.

Ziefle, M. (2002). The influence of user expertise and phone complexity on performance, ease of use and learnability of different mobile phones. *Behaviour and Information Technology - BIT*, *21*(5), 303-311.

Zielinski, D. (2003). Power has always been the point. *Presentations Magazine*, pp. 2,4,6. VNU Business Media.

Züllighoven, H. (2004). *Object-Oriented Construction Handbook*. Dpunkt Verlag.

Züllighoven, H., Bäumer, D., & Bleek, W.-G. (1998). *Das objektorientierte Konstruktionshandbuch. Nach dem Werkzeug- und Materialansatz*. Dpunkt Verlag.

10 List of Illustrations