Acquiring Architecture Knowledge for Technology Design Decisions

Dissertation with the aim of achieving the doctoral degree of

Doktor der Naturwissenschaften (Dr. rer. nat.)

Department of Informatics
Faculty of Mathematics, Informatics, and Natural Sciences
University of Hamburg

Mohamed Aboubakr Mohamed Soliman

Hamburg, 2019

Date of Oral Defense: 2nd of April 2019

The following evaluators recommend the admission of the dissertation:

Prof. Matthias Riebisch, University of Hamburg, Germany

Prof. Reinhold Plösch, Johannes Kepler University of Linz, Austria

Prof. Michel Chaudron, University of Gothenburg, Sweden

Abstract

Software became an essential and important part of our life. A useful software with a high quality has bigger influence on our daily activity. Architectural design decisions have a big influence on the quality of a software system, and are difficult to change once implemented. Software architecture is developed as a result of selecting several software architectural solutions. However, the complexity, diversity and evolution nature of architectural solutions force software engineers to make critical design decisions based only on their own experience. This could lead to wrong or sub-optimal design decisions. In order to support software engineers to take the right design decisions, assistance is required for exploring the architectural knowledge, which encompasses various architectural solutions, their relationships and distinctions.

Technology design decisions (e.g. selecting a framework or a database management system) are one of the most frequently occurring types of architectural decisions, which impact several quality attributes of a software architecture. In the past decades, the number of available technology options has increased significantly and changed constantly, which make technology decisions harder to take. Current architecture knowledge management approaches try to support architects by offering a rich base of architectural solutions, decision factors, and rules. However, current architecture knowledge management approaches depend on manually capturing knowledge about decisions and architectural solutions. Obtaining and evaluating the quality of relevant and reusable knowledge (and ensuring that this knowledge is up-to-date) requires significant manual maintenance effort, which is not feasible with the fast pace of advance in technological development.

The overall problem addressed in this dissertation is to *facilitate acquiring relevant architectural knowledge to take technology design decisions*. We addressed this problem by achieving three main goals: First, we understand technology design decisions in practice. This ensures the practicability of our knowledge capturing solutions. Second, we explore developer communities for architecture knowledge. Developer communities contain an enormous amount of discussions between software engineers, which could be a reliable and an effective source of architecture knowledge. Third, we propose solutions to effectively search for relevant architectural information in developer communities.

To understand technology design decisions in practice, we interviewed practitioners and asked them about their decision making process. Based on the interviews with practitioners, we found that software engineers consider certain technology aspects (e.g. performance of a certain features, documentation and community support) as their main factor to take technology design decisions. These aspects differentiate technologies from each other. While technology vendors describe technology solutions as a set of features, software engineers discuss the differences between technologies in developer communities. As a result from our interviews, we extended existing architecture knowledge models with additional concepts for technology design decisions. The model characterize technology design decisions, and their reasoning.

Current approaches for architecture knowledge management depend on knowledge repositories, which requires much effort to gather, populate and maintain knowledge. Therefore, we explored the most popular online software development community (Stack Overflow) as a possible source of technology-related architectural knowledge. We analyzed posts in Stack Overflow to explore the

type of discussed architectural questions and solutions. We found different types of architecture-relevant posts, which identify and evaluate technology features and components design. To ensure that we explore relevant architecture-relevant posts, we evaluated our classification of posts through feedback from practitioners to create the first corpus of architecture-relevant posts in developer communities, and the first ontology for architecture knowledge in developer communities. The ontology specifies the structure and textual representation of architecture knowledge concepts in developer communities. Moreover, the ontology is empirically grounded through qualitative analyses of different Stack Overflow posts, as well as inter-coder reliability tests.

To facilitate searching for architectural information in developer communities. We developed and compared in a series of experiments a set of classification approaches to identify and classify architecture-relevant posts on Stack Overflow. The classification approaches rely on our proposed ontology of architectural concepts and therefore allow capturing semantic information in Stack Overflow posts rather than only relying on keyword matching and lexical information. Furthermore, we present a novel approach to search for architecture-relevant information in Stack Overflow posts. We implemented this approach in a web-based search engine and compared its effectiveness to a pure keyword-based search in a series of tasks given to practitioners. Our results show that the effectiveness of searching has been improved significantly compared to a conventional keyword-based search.

Kurzfassung

In der heutigen Zeit spielt Software eine wichtige Rolle unser Leben. Eine nützliche Software mit guter Qualität hat einen großen Einfluss auf unsere tägliche Arbeit. Dabei haben Architekturentscheidungen einen großen Einfluss auf die Qualität eines Softwaresystems und sind nach der Implementierung schwer zu ändern. Die Softwarearchitektur ist das Ergebnis der Auswahl verschiedener Softwarearchitekturlösungen entwickelt. Die Komplexität, Vielfalt und Evolution von Architekturlösungen zwingt Softwareingenieure jedoch dazu, kritische Entwurfsentscheidungen nur aufgrund ihrer eigenen Erfahrung zu treffen. Dies führt häufig zu falschen oder suboptimalen Entwurfsentscheidungen. Damit Softwareingenieure unterstützt werden, die richtigen Entwurfsentscheidungen zu treffen, ist es erforderlich, das Architekturwissen, das verschiedene Architekturlösungen, ihre Beziehungen und Unterscheidungen umfasst, zu erfassen.

Entscheidungen zur Technologieauswahl (z. B. Auswahl eines Frameworks oder eines Datenbankverwaltungssystems) sind eine der am häufigsten auftretenden Arten von Architekturentscheidungen, die sich auf mehrere Qualitätsattribute einer Softwarearchitektur auswirken. In den vergangenen Jahrzehnten hat die Anzahl der verfügbaren Technologieoptionen erheblich zugenommen und sich ständig verändert, was technologische Entscheidungen schwieriger macht. Aktuelle Ansätze des Architekturwissensmanagements versuchen, Architekten zu unterstützen, indem sie eine reichhaltige Basis an Architekturlösungen, Entscheidungsfaktoren und Regeln anbieten. Diese Ansätze des Architekturwissensmanagements hängen jedoch von der manuellen Erfassung von Wissen über Entscheidungen und Architekturlösungen ab. Die Qualität von relevantem und wiederverwendbarem Wissen zu erhalten und zu bewerten erfordert einen erheblichen manuellen Wartungsaufwand, der mit dem schnellen Fortschritt der Änderung von Technologien nur schwer umsetzbar ist.

Das in dieser Dissertation behandelte Gesamtproblem besteht darin, die Erfassung von relevantem Architekturwissen zu erleichtern, um Entscheidungen zum Technologieentwurf treffen zu können. Wir haben dieses Problem gelöst, indem wir drei Hauptziele erreicht haben: Erstens verstehen wir die Technologieentscheidungen in der Praxis. Dies sichert die Durchführbarkeit unserer Lösungen zur Wissensgewinnung. Zweitens erkunden wir Online-Developer-Communities nach Architekturwissen. Online-Developer-Communities enthalten eine enorme Anzahl von Diskussionen zwischen Softwareentwicklern, die eine zuverlässige und effektive Quelle für Architekturwissen darstellen könnten. Drittens schlagen wir Lösungen vor, damit der Softwareingenieur effektiv nach relevanten Architekturinformationen in Entwicklergemeinschaften suchen kann.

Um Technologieentscheidungen in der Praxis zu verstehen, haben wir erfahrene Softwareingenieure zu ihrem Entscheidungsprozess befragt. Basierend auf den Interviews lässt sich feststellen, dass Softwareingenieure bestimmte Technologieaspekte (z. B. die Leistung bestimmter Features, Dokumentation und Community-Unterstützung) als ihren Hauptfaktor betrachten, damit sie Entscheidungen zum Softwareentwurf treffen können. Diese Aspekte unterscheiden Technologien voneinander. Während Technologieanbieter Technologielösungen als eine Reihe von Funktionen beschreiben, diskutieren Softwareingenieure die Unterschiede zwischen Technologien in Entwicklergemeinschaften. Als Ergebnis unserer Interviews haben wir bestehende Architekturwissensmodelle mit zusätzlichen Konzepten für Technologieentscheidungen erweitert. Das Modell kennzeichnet Technologieentscheidungen und ihre Argumentation.

Aktuelle Ansätze für das Management von Architekturwissen hängen von Wissensbasen ab, die viel Aufwand erfordern, um Wissen zu sammeln, zu füllen und zu pflegen. Daher haben wir die bekannteste Online-Community (Stack Overflow) als mögliche Quelle für technologiebezogenes Architekturwissen untersucht. Wir analysierten Beiträge in Stack Overflow, um die Art der diskutierten architekturrelevanten Fragen und Lösungen zu untersuchen. Wir haben verschiedene Arten von architekturrelevanten Beiträgen gefunden, die das Design von Technologien und Komponenten identifizieren und bewerten. Damit wir sicher stellen können, dass wir relevante architekturrelevante Beiträge untersucht haben, haben wir unsere Klassifizierung von Beiträgen durch Feedback von Softwareingenieuren mit praktischer Erfahrung evaluiert, um den ersten Korpus von architekturrelevanten Beiträgen und die erste Ontologie für Architekturwissen in Entwicklergemeinschaften zu erstellen. Die Ontologie legt die Struktur und textuelle Darstellung von Architekturwissenskonzepten in Entwicklergemeinschaften fest. Darüber hinaus wird die Ontologie empirisch durch qualitative Analysen verschiedener Stack Overflow Beiträge sowie durch Intercoder-Zuverlässigkeitstests begründet.

Zur Erleichterung der Suche nach Architekturinformationen in Online-Community haben wir in einer Reihe von Experimenten Klassifizierungsansätze entwickelt und verglichen, um architekturrelevante Beiträge auf Stack Overflow zu identifizieren und zu klassifizieren. Die Klassifizierungsansätze stützen sich auf unsere vorgeschlagene Ontologie von Architekturkonzepten und ermöglichen daher die Erfassung von semantischen Informationen in Stack Overflow, damit sich der Softwareingenieur nur auf das Keyword-Matching und lexikalischen Informationen verlassen muss. Darüber hinaus stellen wir einen neuartigen Ansatz zur Suche nach architekturrelevanten Informationen in Stack Overflow vor. Wir haben diesen Ansatz in einer webbasierten Suchmaschine implementiert und deren Effektivität mit einer reinen Keyword-basierten Suche in einer Reihe von Softwareentwurfsaufgaben verglichen, die an Softwareingenieure gegeben wurden. Unsere Ergebnisse zeigen, dass die Effektivität der Suche im Vergleich zu einer herkömmlichen Keyword-basierten Suche einen deutlichen Fortschritt zeigt.

Acknowledgement

Striving to achieve my PhD degree was a long and challenging journey, which would not have been possible without the support from many people. In this page, I would like to thank all, who supported me during my PhD.

First, I would like to thank my supervisor Prof. Matthias Riebisch for giving me the chance to conduct research as part of his research group and under his supervision. I would like to express my deepest gratitude for his guidance, advices and encouragement. His support ensured an excellent working environment and the required motivation to conduct research. Second, I am eternally grateful to the reviewers of the dissertation. Their efforts to read and evaluate the dissertation is quite appreciated. Their comments and feedback will guide me in my future research.

Starting with my PhD would have not been possible without the support from my family. I appreciate my parents (Abou-bakr and Heba), who cared about my education since I was a child. Moreover, I would like to thank my wife Marion and my children Noah and Elias for their love and tolerance. My words cannot describe how grateful I am for their love and sympathy.

I would like to thank the German Academic Exchange Service (DAAD) for their commitment to financially support my PhD. Their support gave me the chance to dedicate my efforts to the research. Moreover, I truly appreciate the participation of more than 30 practitioners during the different phases of the PhD research. They volunteered and gave from their time and effort without any financial rewards. A special thank you to Matthias Galster, Amr Salama, and Olaf Zimmemrann for their research collaboration.

Last but not least, I would like to thank my colleagues in the university of Hamburg; Sebastian, Sandra, Tillman, Yibo, Stephanie, Steffi, and Paula. I appreciate their support, and their collaboration in mutual research projects. It was my honor to work with you in a single research group.

Contents

Contents vii

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Research Study Design . . . 4
  - 1.3.1 Goals of the Study . . . 4
  - 1.3.2 Significance of the Study . . . 5
  - 1.3.3 Research Questions and Contributions . . . 6
    - 1.3.3.1 1st Goal: Understand technology design decisions . . . 6
    - 1.3.3.2 2nd Goal: Explore developer communities for architecture knowledge . . . 9
    - 1.3.3.3 3rd Goal: Propose approaches to search for architectural information in developer communities . . . 11
- 1.4 Thesis Outline . . . 13
- 1.5 Related Publications . . . 15

I Fundamentals of Architecture Knowledge 17

2 Background and Related Work 18
- 2.1 Background on Knowledge Management . . . 18
  - 2.1.1 Characteristics of Knowledge . . . 19
  - 2.1.2 Knowledge Management Process . . . 20
  - 2.1.3 Knowledge Management in Communities . . . 21
    - 2.1.3.1 Overview on Stack Overflow . . . 22
- 2.2 Software Architecture Design Methods . . . 23
- 2.3 Pattern Languages . . . 24
- 2.4 Architecture Knowledge Management . . . 25
  - 2.4.1 Architecture Knowledge Models . . . 25
  - 2.4.2 Architecture Knowledge Sharing . . . 27
  - 2.4.3 Architecture Knowledge Capturing . . . 29
- 2.5 Developer Communities in Software Engineering Research . . . 30
  - 2.5.1 Analyzing Developer Communities . . . 30
  - 2.5.2 Using Developer Communities in Software Development . . . 31
- 2.6 Summary . . . 33

3 Technology Design Decisions 34
- 3.1 Research Question and Contributions . . . 34
- 3.2 Research Process . . . 35
  - 3.2.1 Content Analysis . . . 35
  - 3.2.2 Interviews . . . 37
- 3.3 Technology Features . . . 38
- 3.4 Architecturally Significant Technology Aspects . . . 41
- 3.5 Architecture Knowledge for Technology Design Decision . . . 44
  - 3.5.1 Elements of Architectural Knowledge . . . 44
  - 3.5.2 Solutions' Interactions within an Architectural Knowledge . . . 47
- 3.6 Evaluation of Concepts . . . 48

3.6.1 Interview Responses Analysis Results and Observations . . . 48
3.7 Discussion . . . 50
3.7.1 Interpretation of Results . . . 50
3.7.2 Implications of Results on further Research . . . 51

II Analysis of Architecture Knowledge in Developer Communities 52

4 Architecture-relevant Posts in Developer Communities 53
4.1 Research Questions and Contributions . . . 53
4.2 Research Process . . . 54
4.2.1 Phase 1: Prepare Stack Overflow Posts for Analysis . . . 55
4.2.1.1 Phase 1a: Query for Candidate Posts (Sampling) . . . 55
4.2.1.2 Phase 1b: Initial Manual Classification of ARPs and Programming Posts . . . 56
4.2.2 Phase 2: Classification of ARPs . . . 56
4.2.3 Phase 3: Obtain Feedback from Practitioners . . . 56
4.2.3.1 Phase 3a: Practitioners Selection . . . 56
4.2.3.2 Phase 3b: Evaluation Posts Sampling . . . 58
4.2.3.3 Phase 3c: Evaluation Execution . . . 58
4.3 Types and Variations of Architecture-relevant Posts . . . 58
4.3.1 Types of Architecture-Relevant Posts . . . 58
4.3.1.1 Compact Types of Architecure-relevant Posts . . . 61
4.3.2 Examples for Architecure-relevant Posts . . . 61
4.3.3 Variations from Types of Architecture-relevant Posts . . . 65
4.3.3.1 Variants of Solution Identification ARPs: . . . 65
4.3.3.2 Variants of Solution Evaluation ARPs: . . . 67
4.4 Practitioner Evaluation . . . 71
4.4.1 Agreement and Confidence among Participants . . . 73
4.4.2 Agreement across ARP Types . . . 74
4.5 Discussion . . . 74
4.5.1 Interpretation of Results . . . 74
4.5.2 Implications of Results on further Research . . . 76

5 Architecture Knowledge Ontology in Developer Communities 78
5.1 Research Question and Contributions . . . 78
5.2 Research Process to Define Ontology . . . 79
5.2.1 Data Gathering . . . 79
5.2.2 Data Analysis . . . 79
5.3 Architecture Knowledge Ontology in Developer Communities . . . 82
5.3.1 Simple AK and Lexical Triggers Ontology Classes . . . 82
5.3.2 Composite Ontology Classes . . . 82
5.3.2.1 (CONF) Architecture Configuration . . . 82
5.3.2.2 (CB) Component Behavior . . . 84
5.3.2.3 (EX) Existing System . . . 84
5.3.2.4 (DI) Design Issue . . . 85
5.3.2.5 (REQ) Requirements and Constraints . . . 86
5.3.2.6 (UR) User Request . . . 88
5.3.2.7 (FEAT) Technology Feature . . . 89
5.3.2.8 (ASTA) Technology Benefits and Drawbacks . . . 89
5.3.2.9 (ADD) Recommended design decision . . . 91

5.3.2.10 (DR) Decision Rule . . . 91
5.3.2.11 (CASE) Technology Use-Cases . . . 92
5.3.3 Relationships between Ontology Classes . . . 92
5.3.3.1 Ontology Classes in Post Structure . . . 95
5.3.3.2 Significant Relationships between Ontology Classes . . . 95
5.3.4 Distribution of Ontology Classes in Types of ARPs . . . 97
5.4 Discussion . . . 98
5.4.1 Interpretation of Results . . . 98
5.4.2 Implications of Results on further Research . . . 100

III Solutions for Architecture Knowledge Acquisition 101

6 Scenarios and Perspectives for Using Developer Communities during Architecture Design 102
6.1 Research Questions and Contributions . . . 102
6.2 Research Process . . . 103
6.2.1 Interview Study . . . 103
6.2.2 Analysis of Scenarios . . . 104
6.3 Perspective of Practitioners on AK Sharing and Reuse from Developer Communities 105
6.3.1 Benefits of Using Developer Communities as a Source of AK . . . 105
6.3.2 Problems Faced when Searching for AK in Developer Communities . . . 106
6.3.3 Solutions to Improve the Search for AK . . . 107
6.4 Scenarios for Searching AK in Developer Communities . . . 109
6.4.1 Conceptual Elements of Scenarios to Search for AK . . . 109
6.4.2 Examples and Types of Scenarios for Searching AK in Developer Communities . . . 110
6.5 Discussion . . . 112
6.5.1 Interpretation of Results . . . 112
6.5.2 Implications of Results on further Research . . . 112

7 Classification Approaches for Architecture-relevant Posts 114
7.1 Research Questions and Contributions . . . 114
7.2 Corpus to Evaluate Classification Approaches . . . 115
7.2.1 Development Sample . . . 115
7.2.2 Testing Sample . . . 116
7.3 Exploring Tags for Identifying Architecture-relevant Posts . . . 116
7.4 Classifying Stack Overflow Posts for Architectural Relevance . . . 117
7.4.1 Overview . . . 117
7.4.2 Bag-of-Words Classification . . . 118
7.4.3 Ontology-Based Classification . . . 118
7.4.3.1 Single-ontology-class Classification . . . 119
7.4.3.2 Multi-ontology-class Classification . . . 120
7.4.4 Ensemble Learning . . . 120
7.5 Evaluation of Classification Approaches . . . 121
7.5.1 Accuracy of Classification Approaches . . . 121
7.5.1.1 Study Design . . . 121
7.5.1.2 Results . . . 121
7.5.1.3 Analysis of Classification Accuracy . . . 124
7.5.2 Generalisability of Classification Approaches . . . 125
7.5.2.1 Study Design . . . 125

7.5.2.2 Results . . . 127
7.6 Significant Terms and Ontology Classes . . . 127
7.6.1 Significant Terms to Identify ARPs . . . 127
7.6.2 Significant Ontology Classes to Identify ARPs . . . 128
7.6.2.1 Simple Ontology Classes and Lexical Triggers . . . 130
7.6.2.2 Sequences of Ontology Classes and Composite Ontology Classes 130
7.7 Discussion . . . 131
7.7.1 Interpretation of Results . . . 131
7.7.2 Implications of Results on further Research . . . 132

8 Enhanced Search Approach for Architectural Information in Developer Communities 133
8.1 Research Question and Contributions . . . 133
8.2 Overview of Enhanced Search Approach . . . 134
8.2.1 Steps of Enhanced Search Approach . . . 134
8.2.2 Relating Types of Architecture-relevant Posts and Design Activities . . . 136
8.2.3 Implementation of Search Approach . . . 137
8.3 Evaluation of Search Approach . . . 137
8.3.1 Experiment Design . . . 138
8.3.1.1 Corpus for Experiment . . . 138
8.3.1.2 Architecture Design Tasks . . . 138
8.3.1.3 Experiment Execution . . . 139
8.3.1.4 Measures of Effectiveness . . . 140
8.3.1.5 Exit Survey . . . 140
8.3.2 Evaluation Results . . . 141
8.3.2.1 Effectiveness of the Search Approach . . . 141
8.3.2.2 Results of the Exit Survey . . . 143
8.4 Discussion . . . 143
8.4.1 Interpretation of Results . . . 143
8.4.2 Implications of Results on further Research . . . 144

IV Conclusion 145

9 Summary, Discussion and Future Work 146
9.1 Summary of the Study . . . 146
9.2 Threats to Validity Assessment . . . 148
9.2.1 Threats to Validity for the Interview Studies . . . 148
9.2.2 Threats to Validity for the Content Analysis Studies . . . 149
9.2.3 Threats to Validity for the Experiments . . . 151
9.3 Discussion of the Contributions . . . 152
9.4 Future Work . . . 154

V Appendices 156

A Stack Overflow Posts Analysis 157
A.1 Technology Specification Resources . . . 157
A.2 Stack Overflow Posts Repository . . . 158
A.2.1 Queries of Posts . . . 158
A.2.2 List of Architecture-relevant Stack Overflow Posts . . . 158

A.2.3 Additional examples of Architecture-relevant Stack Overflow Posts . . . 179
A.3 Architecture Knowledge Ontology in Stack Overflow . . . 189
A.3.1 Coding guide . . . 189
A.3.2 Atlas.ti Snapshots . . . 194
A.3.3 Instances of Ontology classes . . . 195

B Interviews Materials **204**
B.1 Interview Questions to Understand Technology Design Decisions . . . 204
B.2 Interview Questions to Understand the Use of Developer Communities . . . 205

C Experiments Materials and Detailed Results **211**
C.1 List of Stack Overflow Tags . . . 211
C.2 List of Distinctive Keywords and Ontology Classes . . . 215
C.3 Experiment to Search for Architectural Information in Stack Overflow . . . 225
C.3.1 Architecture Searching Tasks . . . 225
C.3.2 Queries to Search for Architecture Information . . . 234
C.3.3 Relevant Posts for Searching Tasks . . . 245

List of Figures **265**

List of Tables **269**

Bibliography **272**

1

Introduction

1.1 **Context** 1
1.2 **Problem Statement** 2
1.3 **Research Study Design** 4
1.4 **Thesis Outline** 13
1.5 **Related Publications** 15

1.1 Context

During the past few decades, software has became a key player in all industries (e.g. banking, transportation, manufacturing). The success of a software system could lead to success of the company. This motivates companies to invest in software. For instance, the global IT spendings was more than 3.4 trillion by the end of 2016[1]. The significant increase in the production of software [Boe06] lead to the development of bigger and more complex software systems. With the growing size and complexity of software, requirements for good quality attributes (e.g. performance, maintainability) [ISO01] became more challenging to achieve. Fulfilling quality attribute requirements could only be achieved and maintained by taking right design decisions.

Software architecture is concerned with taking the principal design decisions, which potentially impose risk and uncertainty on software process and product success. These risks originate from the various unknowns that surround the decision maker during the design process. *Architectural design decisions* (ADDs) [JB05, TMD09] must be identified early in the project lifecycle due to their long-term impact on the system quality, and their tenacious behavior, which makes them quite expensive to change [BCK03]. ADDs affect both structural properties of a system and its quality attributes. Examples of ADDs are decisions about components and connectors of a system (e.g., the decomposition of a system into layers and modules), about their configuration through architectural patterns or reference architecture, as well as decisions about selecting implementation technologies (e.g., Java framework). Once an architectural solution is implemented it is quite difficult to change [BCK12]. This differentiates software architecting from implementation e.g., fixing a bug in a method tends to be less complex than changing a technology or pattern.

When taking ADDs, *architectural knowledge (AK)* plays a crucial role for taking good ADDs. AK of software engineers accumulates in their mind when going through different design experiences, or when learning about previous design experiences from other software engineers or researchers.

[1] https://www.gartner.com/newsroom/id/3482917

The accumulated AK involves all the required knowledge about previously taken ADDs, and the resulted architectural components of a software system [KLV06].

One important concept of AK is architectural solutions [ZKL+09]. The selection of suitable architectural solutions is in the core of synthesizing a software architecture [HKN+07]. In the past two decades, several classes of architectural solutions were captured in the current state of the art. Architectural solutions could be classified into *Conceptual Solutions* and *Technology Solutions* [MHvHA11, ZKL+09]:

- *Conceptual Solutions* are abstract solution for design problems, which could be implemented differently in different contexts. Selecting a conceptual solution influences the system architecture configuration (i.e. the components design of the system), independent of the implementation. The knowledge about conceptual solutions is captured in books and catalouges. For example, patterns (e.g. architectural patterns [BMR+96] and design patterns [GHJV94]), design tactics [BCK12], and reference architectures (e.g. [AZE+07]).
- *Technology Solutions* are concrete solutions, which assist the architect to develop and implement the system. This include frameworks, programming languages, standards and libraries. The knowledge about technology solutions resides in technology specifications, vendor web sites, and developer communities (e.g. Forums, Blogs). *Technology design decisions* are concerned with selecting or configuring technology solutions (e.g. selecting technology features). We call the architecture knowledge required to take technology design decisions "*technology-related architectural knowledge*".

Each class of architectural solutions is concerned with solving different types of design problems. Nevertheless, selecting a combination of architectural solutions is required to develop the system software architecture. Such a diversity nature of solutions represents a challenge within the design process, because each solution has its unique impact on the quality attributes and on the subsequent design decisions. As a consequence, it is arduous to select accurately the appropriate solutions for the different architecture design problems.

Van Heesch et al. [vHA11] reported that most architects in practice rely on their personal experience and do not search for other architectural solutions when solving a design problem. This is due to the effort required to search for suitable solutions in between the enormous amount of interacting solutions, that must be selected in a limited project budget and schedule. This common practical problem requires effective knowledge management approaches to facilitate accessing, structuring and searching for relevant architectural solutions to a design problem.

1.2 Problem Statement

The problem addressed in this dissertation is the following:

How to facilitate the acquisition of architectural knowledge to support taking technology design decisions?

We focused on solving this problem because of its practical importance and complexity:

- *Importance of technology design decisions for software architecture*: Technologies impact the implementation of architectural conceptual solutions [BMR+96, BCK03, GKN15], and

quality attributes (e.g., performance, security). For example, a technology may supports or prevent a client-server architecture or a technology may not support architectural tactics such as authorization, authentication or heartbeat. A recent survey [MW13] on the types of architectural design decisions and their documentation shows that around 25% of a system decisions are executive decisions, and most of them are technology decisions. Moreover, technology decisions as well as structural decisions have been observed as the mostly documented design decisions among other categories of decisions. The survey participants indicated that technology decisions are taken early in the design process, and they are quite hard to be changed during the system implementation. These empirical results indicate and affirm the importance of technology design decisions.

- *Complexity of acquiring technology-related architectural knowledge*: Within the past decades, the selection of the right technology product have turned out to be gradually complicated, due to the significant increase in the production of software products by technology vendors, as well as the open source community [Boe06]. Moreover, today's markets are competitive and moving fast with a high rate of innovation [Bos16]. Thus, software engineers have to deal with new technologies, which increase the required efforts to achieve proficiency [IM02]. With an ever increasing number of software technologies [BTH14] and a half-life of software engineering knowledge of five years [Kru15], keeping up to date with technologies is challenging.

Despite the well known significance of technology solutions, the current methods for software architecture design (e.g., [BCK03, Bos00]), as well as pattern languages (e.g. [AZ05]) do not support the architect in making a choice among different technology solutions. Alternatively, they focus on selecting a combination of conceptual solutions as first class concepts for solving the different design issues, presupposing a direct mapping from the conceptual solutions to the technology solutions [CVEK13].

To facilitate acquiring AK about technology solutions for software engineers, researchers and practitioners proposed approaches for codifying (i.e., documenting) and managing AK. This allows architects to capture, share and reuse AK within and across different projects [vHAH12]. One way to capture and share AK is through repositories (e.g. [ZKL+09, GKN15]), which provide catalogs of architectural solutions. AK repositories are based on models, which describe the relationships between different AK concepts. Design issues, architectural solutions, and design decisions are main building blocks of architecture knowledge models.

AK repositories guide architects during the decision making process, and assist software engineers in exploring the design space by browsing and learning new architectural solutions. However, current approaches for AK repositories have the following limitations:

- The conceptual elements of existing AK models do not support choosing suitable technology solutions among several alternatives to a certain architecture design problem. Current AK models model the relationships between technology and conceptual solutions (see Chapter 2 for details), without considering the differences between solutions regarding their capabilities and influences on quality attributes.
- AK repositories rely on *manually* capturing and curating architecture-relevant information [GXY+17]. Due to the effort required to manually collect, and codify AK, architecture knowledge repositories tend to be incomplete.
- AK repositories require significant maintenance to remain useful and up-to-date [vK12],

since technologies evolve constantly [BTH14]. Otherwise, manually captured knowledge in repositories becomes out of date quickly.

Because of these limitations, most organizations do not use systematic AK management approaches [CJT+16]. Rather, software architects exchange experiences through face to face discussions or using channels such as vendor and online developer communities websites [GXY+17].

1.3 Research Study Design

Our strategy for solving the dissertation's problem involve two main targets.

1. Gaining more knowledge about the problem and possible solutions. This could be reached by asking knowledge research questions (e.g. What are the facts?).
2. Changing the environment by proposing methods to address a practical problem. This could be reached by asking practical research questions (e.g. How to improve something?).

We designed our research study to include both knowledge and practical research questions as recommended by existing research frameworks (e.g. design science [Wie09]). Our research questions are grouped and organized based on their goal, which could target either acquiring scientific knowledge or proposing methods to change the environment. The goals of the study are presented in Section 1.3.1 and their associated research questions are presented in Section 1.3.3.

1.3.1 Goals of the Study

The dissertation has three main goals to address the problem statement. The first two goals target gaining new scientific knowledge about technology-related architecture knowledge, while the third goal targets proposing new methods to practically support improving the acquisition architecture knowledge. The three goals of the study are presented below.

1. *Understand technology design decisions*: Because existing approaches for architecture knowledge management do not explicitly and sufficiently model and support technology design decisions, we identify main architecture knowledge concepts for a technology decision. These concepts are important to be determined and understood before proposing methods for knowledge acquisition.
2. *Explore developer communities for architecture knowledge*: Due to the effort required to manually capture and maintain AK (the 2nd and 3rd limitations of existing AK management approaches in Section 1.2), we explored developer communities as a viable alternative to traditional centrally managed knowledge repositories [vK12].

   Why we decided on developer communities? Online developer community web sites (e.g., blogs [PM13], technology forums and Q&A pages such as Stack Overflow) allow software developers to ask and answer questions about software development problems [ASR17]. One of the main success factors of such communities is that they provide a "social motivation" to not only ask questions (i.e., to *benefit* from the community), but also to help others by answering questions (i.e., to *contribute* to the community). By contributing to

a community, users can build up their reputation by being rated based on the amount and the quality of their contribution. Furthermore, these communities provide useful knowledge management features, such as the assessment of user's expertise (e.g., number of "stars" awarded to a contributor), the quality of answers (through community-based voting mechanisms), and the continuous evolution (and update) of knowledge when new questions and answers are added. The benefits provided by developer communities of "social motivation", "community-driven" quality assurance and continuous evolution of knowledge could complement existing AK repositories and AK management approaches.

Experiences from experts in developer communities support software engineers with important technology-related AK which is not provided by technology vendors or architectural documents [dGLT+14b] (e.g., differences between technologies, their benefits and drawbacks). In addition, recent studies (e.g. [PM13]) on social media in software development show that developers discuss high-level concepts, such as features and domain concepts (which are architecture-relevant), as well as topics related to source code. These findings indicate that AK may exist within software development communities. This technology-related AK could be extracted, captured, and prepared for reuse.

3. *Propose approaches to search for architectural information in developer communities*: Developer communities have traditionally been used by developers to solve *coding-related problems* [TBS11a], for example "How to get the cookies from a php curl into a variable"[1]. These questions are often not relevant to architects, because they focus on lower level implementation details. Nevertheless, architects may also benefit from developer communities to solve *architectural problems*. An example of an architecture-relevant post is "What are the benefits and trade-offs of using MSMQ over a SQL Table"[2]. However, the knowledge in developer communities is in the form of unstructured text, which cannot be directly queried for architecture-relevant information. To overcome this problem, we propose approaches to classify and search for relevant architectural information in developer communities.

1.3.2 Significance of the Study

The significance of the study is paramount towards developing practical and useful approaches for AK reuse. Capilla et al. [CJT+16] showed that the reuse features in existing AK management approaches are not well supported and are hard to achieve, because of the complexity of gathering and capturing design decisions. In this study, we argue that using developer communities could provide a practical approach for sharing and reusing AK about technology solutions, because developer communities is the main practical place where software engineers currently share their knowledge.

The success of developer communities to overcome the limitation of traditional knowledge management approaches regarding AK capturing make it promising for knowledge sharing. With the growing trend of using natural language processing (NLP), and information retrieval approaches, they provide possible solutions to overcome the limitation of acquiring knowledge from the unstructured text in developer communities. In this study, we provide the first approach for a specialized search engine to search for architecture information in developer communities. The result of this study maybe used to develop further tools for acquiring and reusing AK from unstructured text. For example, software architects would benefit from a specialized web search engine for software architecture. Moreover, the ability to gather a bigger amount of architectural information

[1] http://stackoverflow.com/questions/895786
[2] http://stackoverflow.com/questions/380052

will support developing further approaches for AK reuse.

1.3.3 Research Questions and Contributions

Fig. 1.1 summarizes research questions for each goal of the study. In addition, it shows our contributions and their relationships to the research questions. To understand the overall research methodology and the relationships between the research questions, Fig. 1.2 shows a flowchart for the overall research processes.

The research questions and contributions for each of the study goals will be discussed in the following sub-sections. We describe the motivation behind each research question, and a brief description on the used research methodology. The details of the research methodologies are presented in the following chapters. Finally, we list our contributions for each research question and goal.

1.3.3.1 1st Goal: Understand technology design decisions

To achieve this goal, we understood first the current state of the practice for taking technology design decisions. After this, we integrated our newly identified concepts for technology design decisions with existing AK models to achieve a comprehensive understanding for technology decisions in context of other AK concepts. In details, we addressed the following two research questions:

- ***RQ1****: How do software engineers conceive software technologies as architectural solutions during the decision making process?*

  Before developing methods to support design decision making, understanding the current state of the practice for architectural design decision is essential. By answering this question, we identify the main concepts for choosing technology solutions during architecture design based on empirical evidence.

- ***RQ2****: How can we model and relate technology decisions with existing architectural knowledge concepts?*

  While several models of AK have been proposed in the current state of the art, it is useful for managing architectural knowledge to integrate and relate existing AK concepts with the newly determined concepts to take technology design decisions.

Research method: We answered RQ1 and RQ2 by conducting an exploratory study (see Chapter 3). We started with a qualitative content analysis for different types of resources, followed by refinement and validation interviews with 7 software engineers. We analyzed the different perspectives, which technology vendors and architects have in offering and choosing technology solutions respectively.

Contributions

The contribution of answering RQ1 and RQ2 is modeling technology solutions as a set of categorized features, which are offered by technology vendors, as well as their associated architecturally significant technology aspects, which are considered by architects to take technology

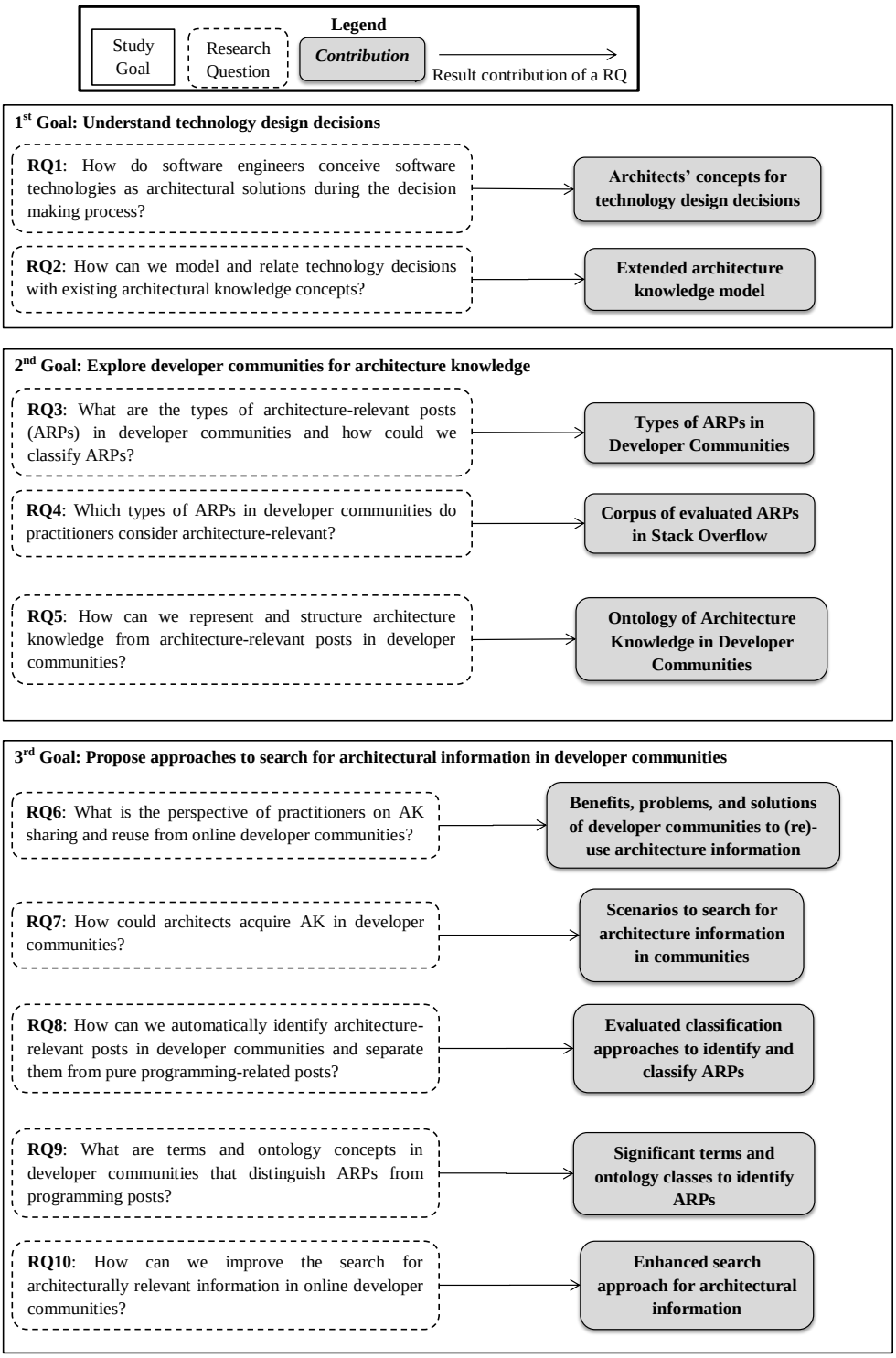


Figure 1.1: Summary of research questions and contributions

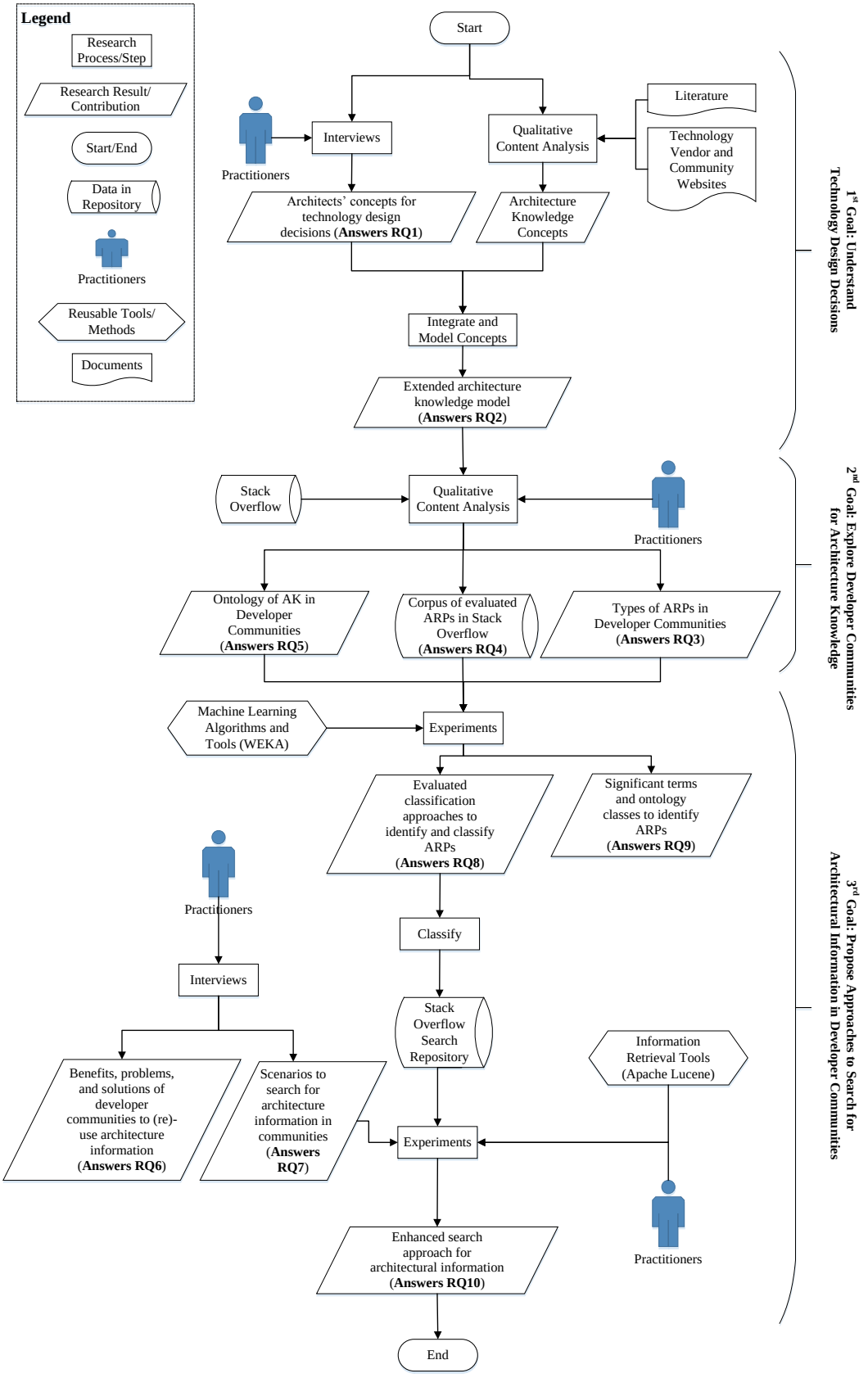


Figure 1.2: High-level overview on the research process

design decision. Finally, we integrated the newly determined AK concepts with existing AK concepts from literature to create an architecture knowledge model, which extends existing AK models.

1.3.3.2 2nd Goal: Explore developer communities for architecture knowledge

In addition to the benefits provided by developer communities (see Section 1.2), our results from answering RQ1 and RQ2 showed that software engineers use developer communities as a source of AK, and specially for technology-related AK. To explore developer communities for architecture knowledge, we conducted a qualitative content analysis study on a sample of posts in a developer community.

To conduct the qualitative content analysis study, we followed two forms of interpretation provided by Mayring [May14]:

- *Summary*: It targets creating a comprehensive overview of the architecture-relevant posts in developer communities.
- *Explication*: It provides additional material about architecture-relevant posts to increase our understanding on existing architecture knowledge in developer communities.

In details, we answered the following three research questions:

- ***RQ3****: What are the types of architecture-relevant posts (ARPs) in developer communities and how could we classify ARPs?*

  Classifying ARPs will allow us to identify different types of AK based on the concerns of stakeholders and architecture decision makers. Exploring types of ARPs facilitate summarizing the AK in ARPs. Also, classifying ARPs into different types will allow us to explore common AK concepts (e.g. design issues, quality attributes) which exist in ARPs. This will enable more targeted approaches for capturing and management of AK (e.g., automated extraction tools of AK from Developer Community Websites). In Chapter 7 and 8, we proposed approaches to facilitate capturing architectural information from developer communities. The proposed approaches depend on the types of ARPs determined from this research question.

- ***RQ4****: Which types of ARPs in developer communities do practitioners consider architecture-relevant?*

  By knowing which types of ARPs practitioners believe are architecture-relevant, we can evaluate our theoretical classifications, and ensures practical relevance of AK extracted from developer communities. Moreover, the evaluation of data by practitioners supports developing a corpus of evaluated ARPs, which have been used to develop automatic classification and search approaches for ARPs (see Chapter 7 and 8).

- ***RQ5:*** *How can we represent and structure architecture knowledge from architecture-relevant posts in developer communities?*

  Answering this question support understanding what AK concepts actually exist in developer communities, and how are they represented in text. The explication of ARPs help to

specify the structure of knowledge in an ARP. This is needed to specify AK concepts in developer communities and to *search* and *capture* AK. For instance, we could better classify ARPs. In addition, this question helps us to further validate whether developer communities are useful not only to answer programming-related questions, but also provides support for architects and "higher level" concerns.

Research method: In order to answer RQ3, RQ4, and RQ5, we selected a sample of Stack Overflow posts, and performed qualitative content analysis to classify ARPs (see Chapters 4). After this, we annotated sentences from a sample of ARPs to identify the representation of AK concepts (see Chapter 5), which has been previously determined from answering RQ1 and RQ2. In addition, we evaluated our assumptions through a feedback evaluation with practitioners.

Why we decided on Stack Overflow as our Sample for a Developer Community? To achieve our 2nd goal, we need an example of an online software development community that provides a good source of well-structured and architecturally related information. Current online developer communities cover a wide range of topics. For example, `http://quora.com` is a generic community for all possible topics, `http://softwareengineering.stackexchange.com` covers topics related to development processes and practices, and `http://stackoverflow.com` covers any topic related to software development and programming. We selected Stack Overflow[1] to gather architecturally relevant information, since it is currently the biggest[2] and most popular online developer community (see [XBL+17]). Also, Gorton et al. recommend Stack Overflow for solving architectural issues [GXY+17]. Moreover, Stack Overflow offer several useful knowledge management features.

1. Stack Overflow follows a Question and Answer (Q&A) structure, which could facilitate categorizing and capturing knowledge.
2. Stack Overflow supports the inclusion of context details in posts (better than Quora), and evaluation of questions and answers through a voting system (better than blogs or forums).
3. The knowledge in Stack Overflow expands and evolves continuously and quickly [BTH14].
4. Stack Overflow provides interfaces for downloading the posts for processing and analysis, which make it easier for us to select a good sample of well evaluated posts (i.e., important problems).

The advantages provided by Stack Overflow motivated us to explore whether (and how) Stack Overflow can be used to extract reusable AK about technology solutions. Our objective was to identify and analyze Stack Overflow posts, which provide useful technology related AK.

Contributions

By answering RQ3, RQ4, and RQ5, we present the following **contributions**:

- Define architecture-relevant posts and classify concrete sub-types for ARPs.
- Evaluate the agreement of practitioners with our classification of ARPs.

[1] `http://stackoverflow.com/`

[2] As of March 14, 2018, stackoverflow.com has 15,508,519 posts (12,812 posts with the "architecture" tag). Other communities have fewer posts. For example, as of Mach 14, 2018, softwareengineering.stackexchange.com has 48,662 posts (2,102 posts with the "architecture" tag.)

- We present the first corpus of evaluated ARPs from a developer community.
- Present an empirically-grounded ontology for AK for one particular community: Stack Overflow. This ontology specifies how each AK concept in ARPs is composed. This bridges the gap between existing theoritical AK concepts and their textual representation in Stack Overflow. The ontology supports annotating and identifying AK concepts within ARPs. Furthermore, the ontology supports ontology-based approaches for searching for AK: Ontology-based approaches specify (e.g., in OWL), capture [WD10] and search [FCL+11] for semantic concepts in text.

1.3.3.3 3rd Goal: Propose approaches to search for architectural information in developer communities

To achieve this goal, we first asked practitioners about the challenges they face to search for architectural information in developer communities. According to the input from practitioners, we focused on improving the effectiveness of searching. We proposed approaches to identify and classify ARPs, as well as an approach to search for architectural information in developer communities. In details, we investigated the following research questions:

- **RQ6:** *What is the perspective of practitioners on AK sharing and reuse from online developer communities?*

  By answering this question, we obtain benefits, problems and solutions from practitioners from their experiences in sharing and reusing AK in developer communities. Practitioners' perspective is important to determine before developing approaches for tools to support acquiring AK from communities. Practitioners' perspectives supported our motivation to determine suitable approaches for AK capturing from developer communities.

- **RQ7:** *How could architects acquire AK in developer communities?*

  Answering this question provides us with concrete use cases of how architects utilize developer communities during architecting. These use cases also offer requirements for community-based tools for knowledge (re-)use during different design activities.

- **RQ8:** *How can we automatically identify and classify architecture-relevant posts in developer communities and separate them from pure programming-related posts?*

  One limitation of developer communities is the unstructured representation of knowledge, and lack of explicit classification of posts between architecture-relevant posts and other types of posts. To overcome these limitations, automatic classification of posts is essential for developing further information retrieval and extraction approaches.

  Some online developer communities allow categorizing questions. For example, Stack Overflow allows to assign related topics to questions in the form of tags. However, most Stack Overflow users assign technology-related tags (e.g. framework names) [TBS11b], and users usually do not indicate what software development life cycle a post relates to (e.g., architecture, coding, testing). Therefore, we cannot directly identify and extract reusable AK based on the "built-in" classification (using tags) of posts. In Chapter 7, we additionally prove that most architecture-relevant posts in Stack Overflow do not have an "architecture" tag.

- **RQ9:** *What are terms and ontology concepts in developer communities that distinguish ARPs from programming posts?*

  Identifying these terms and ontology classes will allow to make a step towards understanding unique characteristics of ARPs to further analyze and capture technology related AK from developer community posts (e.g., automatically separating ARPs from programming posts).

- **RQ10:** *How can we improve the search for architecturally relevant information in online developer communities?*

  Searching for information in unstructured data (such as text in developer communities) is commonly done using keyword-based searches. This research question is motivated by the shortcomings of traditional keyword-based search approaches which cannot deal with the ambiguity of terms for architectural concepts. Gorton et al. [GXY+17] argue that it is challenging for practitioners to create effective search queries for relevant architecture information and internet search engines return many irrelevant results.

  We argue that this is not sufficient to efficiently search for architecture-relevant information. The abstract nature of software architecture concepts makes it difficult for keyword-based searches to find architecture-relevant information: The same architecture concept could be presented with many keywords (i.e., "synonymy") and the same keyword might refer to several concepts (i.e., "polysemy"). For example, if a user executed the query "WCF performance", a keyword-based search will not find Stack Overflow posts which contain the word "throughput", even though throughput is a measure for performance and could be relevant. In addition, some words could occur in different posts with different meanings. For example, the word "server" could be related to an architecture component in an architecture-relevant post, while it could refer to a deployment environment in programming-related posts.

  Therefore, software engineers would beenfit from domain-specific search approaches, which are needed to improve the effectivness of searching over commonly used keyword-based search [XBL+17].

Research method: To answer RQ6 and RQ7, we analyzed literature and conducted interviews with ten practitioners from different companies (see Chapter 6). To answer RQ8, and RQ9, we developed and evaluated using experiments a set of machine learning classification approaches (see Chapter 7). The execution of experiments used our corpus of ARP, which is created from answering RQ3 and RQ4. In addition, some classification approaches use an ontology-based classification approach, which depends on our proposed AK ontology from answering RQ5. To conduct the experiments, we used a reusable machine learning tool (Weka) [HFH+09]. To answer RQ10, we built a search engine (see Chapter 8) using the classification approaches developed when answering RQ8 and RQ9. Moreover, the approach considers ontology concepts in ARPs, which were identified when answering RQ5. We evaluated the effectiveness of the proposed search approach using a controlled experiment with 16 practitioners. The experiment compared our proposed search approach with a normal keyword search. To conduct the experiment, we used a reusable and flexible search engine; Apache Lucene[1]. During the experiment, practitioners solved real architecture tasks, which were created based on our interviews when answering RQ6 and RQ7.

[1] http://lucene.apache.org/

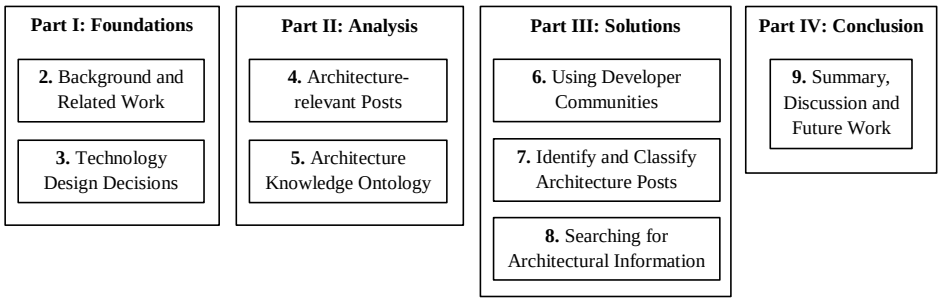


Figure 1.3: Thesis outline

Contributions

In summary, we made the following **contributions**:

- Identify empirically and practically-grounded benefits and problems of, as well as solutions for, utilizing developer communities to share and acquire AK.
- Define concrete use cases for using AK from developer communities during architecture design activities.
- Develop and evaluate a set of *classification approaches* to automatically separate architecture-relevant posts from pure programming-related posts and to further classify architecture relevant posts into sub-categories to support architecture design activities (e.g., evaluation of solution alternatives).
- Identify terms and ontology classes that allow to differentiate ARPs from programming posts.
- Develop and implement a novel approach to search for architecture-relevant posts in developer communities.

1.4 Thesis Outline

The thesis is organized into three main parts, which are divided into several chapters as depicted in Fig. 1.3. Each part is concerned with achieving one goal of the study as previously explained in Section 1.3.1. Part I presents the conceptual foundations for architecture knowledge management. Firstly, It presents a background on knowledge management and an overview on the current state of the art for architecture-knowledge management. Secondly, it presents concepts and models to understand technology design decisions. Part II explains and presents our analysis for developer communities to classify and structure architecture knowledge. Part III proposes solutions to facilitate acquiring architecture knowledge for technology design decisions.

Our contributions are presented in Chapters 3 to 8. Fig. 1.4 shows the contributions in each chapter and their relationships and dependency between each other.

- **Chapter 3** presents architecture knowledge concepts and models for taking technology de-

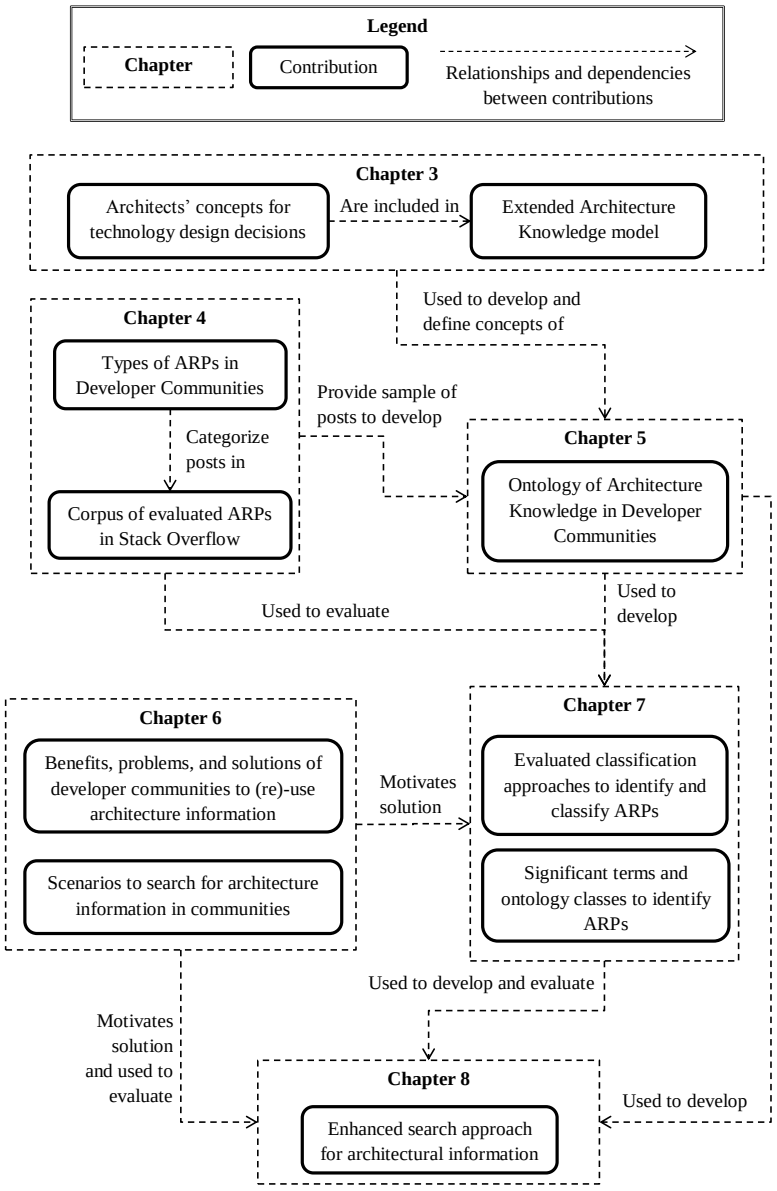


Figure 1.4: Thesis contributions in each chapter and their relationships

sign decisions.

- **Chapter 4** defines architecture-relevant posts and their types in developer communities. Moreover, it presents different types and variations of architecture-relevant posts. The posts are evaluated from practitioners to create the first corpus of architecture-relevant posts from a developer community.
- **Chapter 5** provides a comprehensive explanation for an architecture knowledge ontology in Stack Overflow. The explanation is supported with examples. The development of the ontology depends on posts from the repository of evaluated architecture-relevant posts presented in Chapter 4. Moreover, our content analysis used architecture knowledge concepts for technology design decisions as presented in Chapter 3.
- **Chapter 6** focus on the usage of developer communities by software engineers in the current state of the practice to find architectural information. It presents the problems and scenarios for using developer communities to find architectural information.
- **Chapter 7** provides explanations for the design and experiments of classification approaches to identify and classify architecture-relevant posts in developer communities. The design of some classification approaches depends on our proposed architecture knowledge ontology in Chapter 5. Moreover, the evaluation of the accuracy of classification uses our proposed repository of architecture-relevant posts in Stack Overflow as presented in Chapter 4.
- **Chapter 8** explains the design and experiments for the enhanced search approach to search for architectural information in developer communities. The design of the enhanced search approach uses the ontology and classification approaches from Chapters 5 and 7 to identify and classify suitable architecture-relevant posts for certain architecture design activities. Moreover, the evaluation of the proposed search approach uses the real scenarios provided by practitioners in Chapter 6 to create information seeking tasks, which have been used during the experiments with practitioners.
- **Chapter 9** provides a summary on the overall study. Moreover, it presents the threats to validity for the results. The chapter discusses the contributions of the thesis and provides an overview on possible future work.

1.5 Related Publications

This thesis is based on the following peer-reviewed publications:

- Mohamed Soliman and Matthias Riebisch: Modeling the Interactions between Decisions within Software Architecture Knowledge In: P. Avgeriou and U. Zdun (Eds.): Proc. ECSA 2014, LNCS 8627, Springer, 2014.
- Mohamed Soliman, Matthias Riebisch and Uwe Zdun: Enriching Architecture Knowledge with Technology Design Decisions. In: Proc. 12th Working IEEE / IFIP Conference on Software Architecture WICSA 2015, May 4-7, 2015, Montreal, Canada. IEEE CS 2015. pp 135-144.
- Mohamed Soliman, Matthias Galster, Amr R. Salama and Matthias Riebisch: Architectural Knowledge for Technology Decisions in Developer Communities: An Exploratory Study with StackOverflow. In: Proc. 13th Working IEEE / IFIP International Conference on Software Architecture WICSA2016, April 5-8, 2016, Venice, Italy. IEEE CS 2016

- Mohamed Soliman, Matthias Galster, Matthias Riebisch: Developing an Ontology for Architecture Knowledge from Developer Communities. In Proc. ICSA2017, April 3 - 7, 2017, Gothenborg, Sweden, pp 89-92, 2017
- Mohamed Soliman, Amr R. Salama, Matthias Galster, Olaf Zimmermann, Matthias Riebisch: Improving the search for architecture knowledge in developer communities. In Proc. ICSA 2018, Seattle, Accepted for publication.

Further publications related to the thesis from the author of the thesis:

- Sebastian Gerdes, Mohamed Soliman, and Matthias Riebisch: Decision Buddy: Tool Support for Constraint-Based Design Decisions during System Evolution. In: Proc. 1st International Workshop on Future of Software Architecture Design Assistants (FoSADA '15) May 6th, 2015, Montreal, Canada. ACM, 2015, pp. 13-18.
- Sandra Schroder, Matthias Riebisch, Mohamed Soliman: Architecture Enforcement Concerns and Activities - An Expert Study. In Proc. 10th European Conference on Software Architecture - ECSA 2016, 28 November - 2 December, 2016, Copenhagen, Denmark. Springer, LNCS 9839, pp. 247-262, 2016.
- Sandra Schroder, Mohamed Soliman, Matthias Riebisch: Architecture Enforcement Concerns and Activities - An Expert Study. Journal of Systems and Software, Volume 145, November 2018, Pages 79-97, Elsevier.

Part I

Fundamentals of Architecture Knowledge

2

Background and Related Work

2.1 Background on Knowledge Management 18
2.2 Software Architecture Design Methods 23
2.3 Pattern Languages 24
2.4 Architecture Knowledge Management 25
2.5 Developer Communities in Software Engineering Research 30
2.6 Summary 33

In this chapter, we provide first in Section 2.1 a brief background on existing concepts and methods for knowledge management. We then discuss in Sections 2.2 to 2.5 the most important related work to our contributions in this thesis. We explored related work in the following areas of research:

- In Section 2.2, we explain briefly existing architecture design methods and processes.
- In Section 2.3, we explore patterns and pattern languages and its relationship to architecture knowledge.
- In Section 2.4, we analyze and discuss in details existing approaches for architecture knowledge management. We present existing architecture knowledge modeling, sharing, and capturing approaches.
- In Section 2.5, we give a short overview on using developer communities in software engineering.

2.1 Background on Knowledge Management

This thesis uses and depends on several existing concepts in the field of knowledge management [Dal11]. Thus, we provide in this chapter an overview on the most relevant concepts and methods for knowledge management.

One important motivation for the field of knowledge management is the significant advance of companies, which were able to efficiently and effectively use and create new knowledge [DPP97]. In big companies, knowledge is considered as a valuable asset, which is embedded in companies' products and services (especially technology products and services). To support companies in managing their knowledge, researchers in the field of knowledge management propose approaches

to capture, share and use knowledge. We will give an overview about them in the following subsections.

2.1.1 Characteristics of Knowledge

Knowledge as a concept is one of the most studied topics, which has been researched in several fields such as, epistemology, knowledge management, and computer science. Each field describes different characteristics of knowledge.

At the epistemological level, researchers classified knowledge into five kinds [Aud10]:

1. *Perceptual*: knowledge based on actual perception (i.e. seeing, hearing, feeling). For example, our feeling for a cold glass of water.
2. *Memorial*: knowledge stored in memory about a certain event. For example, our memory about our first experience to ride a bike.
3. *Introspective*: knowledge emerges from an imagination. For example, our imagination of a blue sea.
4. *Priori*: knowledge based on previous knowledge to a certain observational experience. For example, if we know that John is taller than Andy, and Andy is taller than Bruce, then John is taller than Bruce.
5. *Inductive*: knowledge which is created based on a generalization from a previous knowledge. For example, our knowledge that a cactus plant can grow without much water.

In the field of knowledge management, knowledge is distinguished from other concepts like assets, data and information using several characteristics.

First, knowledge distinguish itself from other assets in an organization through four characteristics [Dal11]:

1. Using knowledge does not consume it.
2. By transferring knowledge, we do not lose it.
3. Knowledge is abundant, but the ability to use it is scarce.
4. Controlling and managing knowledge is challenging, because it is tacit in the brain of employees in an organization.

Second, we could distinguish knowledge from data and information through two characteristics [Dal11]:

1. Knowledge is based on individual values, perceptions, and experience. This makes knowledge more subjective compared to information and data.
2. Knowledge could be classified into two types: Tacit and explicit knowledge:

- *Tacit knowledge* exists within the heads of knowers, and is difficult to put into a tangible form like words, text, or drawings. Moreover, Tacit knowledge represents the know-how for producing a certain product.
- *Explicit knowledge* is tangible and could be found in contents like text or photos or videos. Explicit knowledge represents the final-product.

Most knowledge is either tacit or rooted in tacit knowledge [Pol66]. Tacit knowledge represents the hidden part of an iceberg for the entire body of knowledge, while explicit knowledge represents only the tip of the iceberg. The movement of knowledge between tacit and explicit knowledge creates and produces new knowledge. Knowledge is converted between tacit and explicit knowledge in four modes:

1. *Socialization*: This happens when individuals communicate directly with each other (e.g. face to face). In this case, tacit knowledge from one individual is transformed to another tacit knowledge in the mind of another individual.
2. *Externalization*: When writing or recording our knowledge in a tangible form (e.g. text). Tacit knowledge is transformed into explicit knowledge
3. *Combination*: When processing existing explicit knowledge (e.g. in a database) and transform it to another explicit form (e.g. generating a report from a database).
4. *Internalization*: When individuals read or watch existing explicit knowledge, they learn and grasp new knowledge in their mind. Thus, explicit knowledge is transformed into tacit knowledge.

2.1.2 Knowledge Management Process

"Knowledge management is the deliberate and systematic coordination of an organization's people, technology, processes, and organizational structure in order to add value through reuse and innovation. This coordination is achieved through creating, sharing, and applying knowledge as well as through feeding the valuable lessons learned and best practices into corporate memory in order to foster continued organizational learning" [Dal11].

Applying knowledge management involves three levels of application: Individual, community, and organization. Knowledge management at the community and organization levels support individuals with a bigger and extended body of knowledge. The combination between a body of knowledge (i.e. from community and organization) and the innovative skills and competencies of individuals support companies to compete effectively.

Knowledge management has the following features [Dal11]:

- Able to generate new knowledge by motivating individuals.
- Access external knowledge sources.
- Use knowledge to support making decisions.
- Integrate knowledge in products, services and processes.
- Expose knowledge in documents, databases and software.

Researchers in the field of knowledge management proposed several models (e.g. [BW99] and [McE02]) for modeling knowledge management processes. Existing models share common phases for knowledge management [Dal11]:

1. *Knowledge capturing and codification*: In this phase, knowledge is transformed from its tacit to explicit form. Interviews and story telling are two well know knowledge capturing methods.
2. *Knowledge sharing*: Hardware and software technologies (e.g. Networks, databases) support transferring knowledge between individuals. Well known database-management systems as well as wikis are commonly used to facilitate knowledge sharing.
3. *Knowledge application*: Knowledge is used for decision making and problem solving.

2.1.3 Knowledge Management in Communities

Studies on knowledge management in companies show that employees prefer to communicate with other individuals to learn about new knowledge when taking a decision. Thus, learning shows to be a predominantly social process [CP01], where individuals interact with a group of individuals (i.e. a community), who share similar interests. During this process of social interaction, new knowledge emerges in the connection among individuals. Social constructivism views knowledge as a subjective, social artifact. They argue that social interaction produces a type of knowledge, which is a shared understanding among individuals [Dal11].

Currently with new advances in technologies, individuals could virtually communicate with each others and share their knowledge using many types of technologies (e.g. forums, e-mail groups, discussion groups). These groups are referred to as *communities of practice*. A community of practice can be defined as "a group of people, along with their shared resources and dynamic relationships, who assemble to make use of shared knowledge, in order to enhance learning and create a shared value for the group" [AF00].

Once a community of practice grows and expands in its knowledge base, it becomes adopted by individuals and organizations. Virtual communities showed their economic and social significance, because they encourage individuals to continuously interact with each other [HA97]. Communities of practices offer the following benefits for individuals:

- Support individuals to gain reputation.
- Connect individuals together, who could be distantly located.
- Support knowledge sharing and creating new knowledge. This reduces time for innovation and reduces mistakes during work.
- Support creating standardized solutions for problems.

Communities of practice share common characteristics [Wen98]:

- *Joint enterprise*: This is the shared goal, which binds members of a community together. Members of a community are mostly motivated to improve their profession. Employees today are more often loyal to their profession than to a particular company.

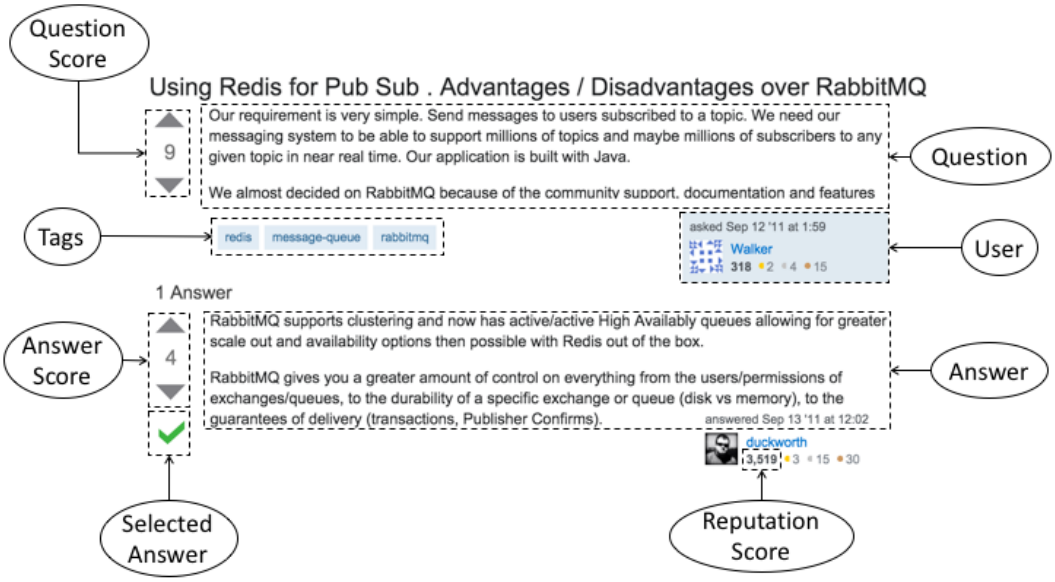


Figure 2.1: The structure of a Stack Overflow post. The dotted rectangles represent regions in a Stack Overflow post. Bubbles specify elements of a post, and their associated region.

- *Mutual engagement*: This is the activities and behavior of members to become part of the community. Most communities are self-regulated. Thus members need to be responsible on following a certain code of ethics.
- *Shared repertoire*: This is a shared virtual work-space, where discussions and artifacts are stored. Several technologies are used for this (e.g. Wikis, web technologies, database management systems).

There are many types of communities of practice, according to its topic. For example, a community about a certain profession or industry (e.g. engineering or automotive), or a community about certain types of technologies. Developer communities are one type of community, which are concerned with software development topics. We will give an overview in the next section on Stack Overflow, which is one example of a developer community.

2.1.3.1 Overview on Stack Overflow

Stack Overflow is a software developer community, which allows software engineers to submit questions and provide answers to questions. The main element, which build the Stack Overflow community is *posts*. A post consists of a question and optionally one or more answers for this question. Each post starts with posing a question from a registered user. The question describes a problem and context details. When creating a question, the user who asked the question can add a list of tags to specify the topic of the post. Other users could read the post and propose answers to the question. The user who asked the question can mark a single answer that answers the question best (according to this user's perception) as the accepted answer.

Stack Overflow supports several additional features to support socialization between users. For example, when posing an answer, users could comment on an answer, which support users to directly discuss with each others. Moreover, the post question as well as each answer have an associated score, depending on the voting inputs from other users. Useful and well-structured

questions receive higher scores than questions that are not relevant or not well written. Each registered user has a reputation score. Posing or answering a question with a high score will increase the reputation of a user. Thus, users are motivated to submit good questions and answers, in order to build better reputations. Fig. 2.1 provides a snap shot for a Stack Overflow post, and shows its main elements.

As of June 2018, Stack Overflow has nearly 16 million posts, 240 thousands users, and 1539 tags. Thus, Stack Overflow with this amount of knowledge is the biggest developer community, which is daily considered by software engineers and organizations.

2.2 Software Architecture Design Methods

In the past few decades, several prominent software architecture design methods (e.g. RUP 4+1 views [Kru95], [HKN+07]) have been suggested and utilized in practice. The proposed methods target modeling the software architecture in several views, such that each view comprises distinctive diagrams for modeling the proposed solution, in order to satisfy the stakeholders' architectural concerns. In addition, the methods provide several guidelines for the correlation between the different viewpoints. Most of the methods dedicate a viewpoint for the implementation or system realization. For example, one of the RUP 4+1 views is the the development view, which model the proposed solution technology components and connectors, providing guidelines for taking the technology decisions, such as ease of development, software management and reuse.

Also researchers propose processes for architecture design (e.g. Attribute Driven Design [KC16]), which describe generic design steps for the activities to develop a software architecture. For instance, the Attribute Driven Design [KC16] propose an iterative process, which consists of 7 steps:

1. *Review Inputs*: Software engineers review their inputs before starting with the design. They ensure to have a clear purpose (e.g. produce a design for an early estimate). They also review the functional requirements, quality attributes and architectural constraints. In addition, software engineers also check existing systems, which will be involved in the architecture.

2. *Establish goals for the design iteration*: An architecture is usually developed in iterations. Each iteration focus on certain goals (e.g. focusing on a certain design issue like establish interoperable communication between two systems). A software engineer needs to specify the goal of an iteration.

3. *Choose one or more elements of the system to refine*: A software architecture is complex and consist of several interrelated components. To produce an architecture, a top-down or bottom-up approach could be used to decompose and combine components of an architecture. In this step, the architect will select a component for further refinement.

4. *Choose design concepts*: This step is the most challenging and the core of taking design decisions. It consists of two sub-steps:

   (a) *Identify design concepts*: Identify suitable types of architectural solutions (e.g., patterns, tactics, families of technologies), and search for candidate solutions (e.g., layers in case of patterns, or RabbitMQ for case of broker technologies).

(b) *Select design concepts*: Compare and select most appropriate solutions. This is done through analyzing benefits and drawback of alternative design concepts, by performing additional analyses and building prototypes (if needed).

5. *Instantiate architecture elements*: Selected design concepts are customized to the design issue. For example, when selecting a layered pattern, the number of layers, components in layers, and their communication need to be decided. Another example is decision on suitable authorization features in a technology if security is a key issue.

6. *Sketch views and record design decisions*: After taking design decisions, software engineers can document his decisions and architectural models in different views (e.g. using RUP 4+1 views [Kru95]).

7. *Perform analysis of current design and review*: To evaluate the design decisions, software engineers discuss their architecture with the stack-holders (e.g. using ATAM). This ensures that they addressed all requirements and constraints.

Analysis and critique
The proposed architectural design methods and processes extensively discuss and describe main activities of design and their sequences. However, they provide minimum support for managing architectural knowledge [FCK07]. Consequently, this makes architects depend on their personal experience, instead of reusing and learning from others experiences.

2.3 Pattern Languages

A pattern language is concerned with defining a group of patterns, which solve related problems in the same domain. Moreover, some pattern languages provide relationships between patterns, to support the architect taking design decisions through moving from one pattern to another. Each sequence of patterns produce different design result. For example, Buschmann et al. [BHS07a] define a pattern language for middelware patterns. One path of decisions inside Buschmann's pattern language is the decision on the communication between layers. Buschmann et al. propose that after selecting the layers pattern, you might select the broker pattern to make layers communicate with each other.

Patterns have several types of relationships with each other [BHS07b]:

- *Alternative*: Context and forces impact the selection of alternative patterns. Part of this context could be previously taken design decisions like selecting a programming language. For example, deciding between request/reply and one way communication patterns.
- *Cooperation*: One pattern could complement another pattern to make it more complete. This usually happens after deciding on the first pattern, which trigger another design issue. This triggered design issue could be solved by the second pattern. For example, after deciding on the layers pattern, pipes and filters pattern could be used to specify the dynamic flow between layers.
- *Combination*: When having two alternative patterns, sometimes it is hard to decide on one of them. In some situations, the designer could combine both patterns together to produce a new pattern, which is a mix of both patterns.

Analysis and critique

In the past few decades, many pattern languages have been proposed (e.g. [BHS07a]). Each pattern language addresses a different domain of problems. Typically, pattern languages do not incorporate technology solutions as first class elements within their network of decisions. Nevertheless, each pattern provides optionally a list of technology examples, which implement this pattern. A proposed pattern language [MHvHA11] integrate technology solutions with pattern languages. The proposed language model interrelationships between different technology solutions, as well as an implementation relationship between technologies and patterns. However, the suggested solution's network does not provide enough guidance for decision making and reasoning on technologies, because of the limited and insufficient modeling elements (e.g. technology features) and relationships (e.g. relationships to design issue) to support technology decisions.

2.4 Architecture Knowledge Management

In this section, we present some of the most important related work in architecture knowledge management, and clarify their differences to our contributions. Existing literature reviews on architecture knowledge [WG14] and architectural design decisions [TGAS14] provide comprehensive list for publications on models, approaches, and tools for architecture knowledge management. In the following sub-sections, we present and discuss existing related work based on their contributions into architecture knowledge models, architecture knowledge sharing, and architecture knowledge capturing.

2.4.1 Architecture Knowledge Models

Jansen et al. [JB05] proposed the first high level conceptual model for the main elements of architecture knowledge. Their model presents a paradigm shift to focus on architectural design decisions as first class entities. They defined architectural design decisions as "A description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture". Their proposed AK model shows the relationships and transformations between architecture models (i.e. components and connectors) and design decisions. The model viewed design decisions from a "problem - solution" point of view; the problem is derived by a motivation and cause such as the architectural significant requirements and stakeholders concerns or other architectural decisions, and a list of architectural solutions are proposed to solve this problem. Each solution has its own characteristics of design rules, pros and cons.

After the work of Jansen et al. [JB05], researchers proposed several architecture knowledge models. Each model propose additional conceptual elements. For example, De Boer et al. [dBFL+07] proposed a conceptual model for main concepts of architecture knowledge. The main contribution of the model is modeling forces, views, and stakeholders, and their relationships to design decisions. Capilla et al. [CZZK11] proposed an enhanced architecture knowledge model, which links knowledge concepts with artifacts such as requirements documents, and source code artifacts. The model also considers the evolution nature of these artifacts and their influence on different architecture knowledge concepts.

Kruchten et al. [KLV06] classified architectural design decisions into 4 kinds: A) Existence decisions, B) Bans or non-existence decisions, C) Property decisions, and D) Executive decisions. The authors also defined several attributes and relationships between the kinds of decisions. For example, two decisions could be alternatives to each other or conflict with each other. Moreover, the authors distinguished between several types of design decisions according to the awareness and documentation of decisions. For example, a decision could be implicitly taken (i.e. the architect is unaware of the decision) and undocumented, or explicit and documented.

The work from Tang et al. focuses on understanding and modeling the reasoning process of taking an architectural design decision. In other words, they try to model and depict the different methods, that the architect can use to think about an architecture design problem. First, Tang et al. [TBGH05] explored in a survey the different types of rationale for a design decision (e.g. design constraints or assumptions). They also explore the different methods and challenges for documenting design rationale. Second, Tang et al. [TTHV08,THV09] proposed a model for architecture design reasoning called AREL. The model relates design decisions with design concerns and outcomes. Moreover, the authors proved based on an experiment that using the proposed reasoning model improves the quality of software design. Moreover, Tang et al. [TVV09] classified and modeled the different types of constraints, which influence architecture design reasoning. The authors grouped constraints into requirement, quality requirement, contextual, and solution constraints. Furthermore, the authors proposed an integrated constraint-based design reasoning process. Recently, Tang et al. [TBSvdW18] propose an approach to improve the reasoning behind taking a design decision. The approach is based on reminder cards, which support architects to remember their decisions, and their rationale. On the same line of research, Falessi et al. [FCKK11] experimented and compared several decision making techniques. The authors provide an approach to choose between the different decision making techniques.

Zimmermann et al. [ZKL+09,Zim11] proposed a model for reusable architecture knowledge. The model separated two concepts, the solution (alternative) and their concerns (i.e. Design issue) and the outcome for an architecture design decision taken inside the project. Moreover, the model groups design issues into groups and levels. The levels of design issues and solutions are conceptual, technology, and products. The authors additionally formalized the relationships between design issues and solutions, and categorized them into three categories: A) Elementary such as "the *alternative* relationship between two solutions". B) Logical such as "taking a decision *forces* taking another decision". C) Temporal such as "deciding on a certain architectural solution *trigger* another design issue". Finally, the authors presented dependency patterns between design decisions. For example, one pattern is "Vendor push", this pattern happens when a vendor makes a constraint on a certain product or technology.

Several researchers proposed models and templates to document architecture design decisions. Tyree at al. [TA05] proposed one of the earliest templates to document a design decision. The template has several properties like status, constraints, and argument. Van Heesch et al. [vHAH12] proposed a framework for documenting architecture design decisions. The framework consists of four views, which model design decisions from different perspectives. The views for a design decision are decision detail view, decision relationship view, decision chronology view, and decision stakeholders view. The decision detail view contains main properties of a design decision such as the design problem, alternatives, and taken design decision. The decision relationship view relates the design decisions with each other. For example a decision could be caused by other decisions, or a decision depends on another decision. The decision chronology view models the flow of decisions and their status change with time. For example, a decision could be first proposed, then decided, and then probably canceled. Finally, the decision stakeholders view models the involvement of stakeholders in each design decision. For example, a stakeholder could confirm or validate

a decision. The architecture decision view has been further used in an industrial setting to support decision space exploration [vHJPB+17].

Researchers proposed specialized architecture knowledge models. Jansen et al. [JdVAvV08] proposed empirically a specialized AK model to support sharing quantitative analysis results. The model could be used additionally to validate design models. Lewis et al. [LLA16] proposed a decision model, which guides the selection of architectural tactics according to the functional and non-functional requirements for cyber-foraging systems.

Gorton et al. [GKN15] modeled the architecture knowledge for big data technologies. The authors classified technology features according to their impact on quality attributes. For example, features for achieving consistency are "ACID transaction" and "Replicas", other features to support scalability are "Scalable distribution architecture" and "load balancing". Moreover, the proposed model relates big data technology features with architectural tactics. For example, the "Read repair" replication technology feature is related to the ant-entropy repair and hinted handoffs tactics.

Analysis and critique

- Researchers proposed several models for architecture knowledge, which consider high-level concepts of architecture knowledge and their relationships. However, most architecture knowledge models (with the exception of [ZKL+09] and Gorton et al. [GKN15]) do not explicitly model technology solutions as a separate first class entity, and therefore, they do not support neither the reasoning on technology design decisions nor the reuse of technology-related architecture knowledge.
- The architecture knowledge model proposed by Zimmermann et al. [ZKL+09] models technology solutions explicitly. Moreover, the authors modeled the relationships between technologies and other solutions (e.g. patterns). However, the proposed AK model lacks the ability to distinguish between different technologies' capabilities (i.e. features) from each other, such that it is hard for a software architect to choose suitable technology solution for the project situation. We believe that the approach proposed by Zimmermann et al. is promising regarding architecture knowledge reuse. Therefore, we extended this model in Chapter 3 to support technology design decisions.
- The architecture knowledge model proposed by Gorton et al. [GKN15] is also promising regarding its support for modeling technology solutions. The model include explicit modeling for technology features and their relationships to architectural tactics. However, the model do not provide any guide regarding the differences between technologies (i.e. the benefits and drawbacks of each technology). This information is important to decide on a certain technology solution.

2.4.2 Architecture Knowledge Sharing

Before developing an architecture knowledge management system, requirements for architecture knowledge management need to be identified. Clerc et al. [CLvV07] determined empirically different use cases for using architecture knowledge management systems. Examples of use-cases are "check implementation against design decision" and "Reuse design decisions". The use cases are modeled and grouped according to stakeholders and functionality. Moreover, several levels of architecture are considered such as software, system, enterprise and processes architectures.

Since the paradigm shift of describing the software architecture as a set of architectural design decisions [JB05], several architecture knowledge sharing tools have been developed to support sharing architecture design decisions. Each tool is based on a different architecture knowledge model and uses different terminologies. In addition, each tool provides different capabilities to assist the architect in dealing with the ADDs. Existing surveys [SLK09, TAJ+10] have been conducted to evaluate and compare the different AK tools. However, each survey uses a different set of evaluation criteria. In order to understand the differences in the capabilities between the different tools, we analyzed and consolidated the different surveys. Moreover, we appended additional recent tools, which have not been considered by the surveys. In the following paragraphs, we analyze and report the differences between the different tools in regarding to their approaches for knowledge sharing.

Several tools used classical web-based technologies and databases to support architecture knowledge sharing. PAKME [BWG05] is a web-based tool, which provides services for architecture knowledge acquisition, maintenance, retrieval and presentation. In addition, the knowledge is classified between project specific and generic. The tool supports a library of generic patterns, which is supported with an advanced searching capability. ADDSS [CNPDn06] is an architecture knowledge sharing tool, which provides the ability to document design decisions for each project iteration, as well as the dependencies between them, through constrains relationships. In addition, the tool provides a repository for patterns, to support the architect selecting an architectural solution. Farenhorst et al. [FILvV08] proposed a portal for sharing architecture knowledge using a just in time approach for knowledge capturing. The portal provides flexible features using plugins, which can customize the environment differently for each organization.

The Wiki technology has been explored to support organizing and populating architecture knowledge. ADkwik [SZP07] is a web based tool, using wiki technologies. The tool provides the ability to capture design decisions through selecting the suitable solution alternative from a list of stored architectural solutions. In addition, it relates the different design decisions using different types of relationships. The tool is based on the model of Zimmermann e*t al.* [ZKL+09], which guides the architect in exploring the design space. De Boer et al. [dBvV11] experimented with using semantic wikis as a platform for architecture knowledge sharing in an industrial environment. The wiki has been used to model and store design decisions on the enterprise level for e-Government software systems. Gorton et al. [GKN15] proposed an architecture knowledge repository for big data technologies. The authors used semantic wikis for storing and modeling knowledge. The wiki is based on a model, which relates technology solutions with technology features and architectural tactics. The wiki provides several useful features such as advanced queries. The query is able to consider different semantic entities in the wiki.

Researchers proposed architecture knowledge sharing tools as *plugins* to software development and modeling tools. Archium [JDAH07] is an eclipse based plugin tool, which provides the ability for the architect to describe the software architecture through a textual description. The Archium compiler visualizes the software architecture components and their associated ADDs. Moreover, it supports consistency check and traceability between the implemented Java code and the architecture design, which supports architects during system implementation and changes. ADvISE [LTZ12] is an eclipse plug-in tool, which support reasoning about design decisions using the QOC (Question, option, criteria) concept. Additionally, it supports documenting the taken design decisions. An extension to the tool [LZ13] supports the uncertainty of taking design decision through fuzzy logic. Moreover, the tool supports relating design decisions to design diagrams to ensure the consistency between design decisions and design diagrams. An industrial implementation for design decisions documentation framework [MTK+14, MTA+16] is proposed as a plugin to the Enterprise Architect modeling platform. The plugin supports documenting the chronologi-

cal order of ADDs. Moreover, the tool supports the traceability between ADDs and the different artifacts within the development process.

A community website for software architecture has been proposed. Software Architecture Warehouse [NP13] is a collaborative decision making web site, which provides different architectural solutions for design issues and gives the ability for different users to collaborate and discuss about solutions. In this way, each proposed solution is evaluated, which supports the decision maker to select the right solution.

Analysis and critique

- A recent survey [TGAS14] on the architectural decisions field shows that most of the architecture knowledge sharing approaches focus on documenting and capturing project specific design decision rationale (e.g. [KLV06], [vHAH12] and [FCK08]) for the sake of minimizing the software architecture erosion phenomena. Moreover, a recent study on the software architecture knowledge [WG14] shows that, the current architecture knowledge approaches have less support regarding architecture knowledge sharing and reuse.
- All proposed approaches depend on populating and gathering architecture knowledge manually, which requires a significant effort to capture and gather properties and relationships between decisions. Due to the amount of effort, approaches for architecture knowledge management are not used in industry [CJT+16] (with limited exception [MTK+14, MTA+16]).

2.4.3 Architecture Knowledge Capturing

Several approaches have been proposed to capture architecture knowledge. Some approaches still relay on manually capturing the knowledge. However, their methods try to facilitate capturing the knowledge. Other approaches automate capturing architecture knowledge.

Due to the effort required to document architecture knowledge manually, approaches have been proposed to facilitate the capturing process. Falessi et al. [FBC+13] proposed an approach for documenting decisions based on categories in decisions which are most relevant to a design activity (e.g. assumptions, alternatives). Miesbauer et al. [MW12] tried to solve the same problem by considering context information of development and design artifacts (e.g., time of creation, relationship with other artifacts) as a source of knowledge during documentation. Tofan et al. [TGA11] proposed an approach for capturing tacit architecture knowledge through interviewing architects using a method known as the Repertory Grid Technique. De Graaf et al. [DGLT+14a] provide a method and guidelines to develop an ontology for architectural knowledge from architecture documentation.

Due to the high effort for architecture knowledge capturing manually, approaches have been proposed to automate architecture knowledge capturing. Van der Ven et al. and Bosch [vdVB13] proposed an approach to automatically analyze and capture information about design decisions from version management data of large open-source repositories. Lopez et al. [LCAC12] and Anvari et al. and Zimmermann [AZ14] proposed methods based on natural language processing and using keywords to capture architectural design decisions from existing text documents. Bhat et al. [BSB+17, BSK+18] proposed a machine learning approach to automatically identify and classify design decisions in issue management systems. Moreover, they used their classification to recommend suitable experts to address certain design issues. Gorton et al. [GXY+17] proposed

an approach based on machine learning to automatically recommend web pages, which contain relevant architectural knowledge related to certain technology features. The approach focused on technology specification documents.

Analysis and critique

Approaches have been proposed to capture tacit and explicit architecture knowledge to support system evolution and design reasoning. While most approaches still rely on manually capturing architecture knowledge, few approaches automate the knowledge capturing process. Automated architecture knowledge capturing approaches considered four sources of architecture knowledge: 1) Issue management, 2) Architecture documents, 3) Source code repositories, and 4) Technology documentation. Nevertheless, existing approaches for architecture knowledge capturing do not analyze or capture architecture knowledge in online developer communities. As previously explained in Chapter 1 and in Section 2.1.3, developer communities is currently the place where software engineers discuss and share their experience about technologies. Moreover, developer communities provide several useful knowledge management features (e.g. questions and answers evaluation), which are not provided in other sources of knowledge. Therefore, we believe that developer communities are important and useful source of architecture knowledge, which worth to be analyzed and reused. In Chapters 4 and 5, we analyze developer communities for architecture knowledge, and in Chapters 6, 7, and 8, we propose approaches to support capturing architecture knowledge in developer communities.

2.5 Developer Communities in Software Engineering Research

Developer communities like Stack Overflow have been researched to support software engineers in conducting different software development tasks. Current approaches did not consider developer communities for architecture tasks. However, they used developer communities to support programming activities. We provide a brief overview on the most well-known and recent related work in this area of research.

2.5.1 Analyzing Developer Communities

Stack Overflow has been a subject to analyze programming posts and behavior of developers.

1. *Analyzing programming posts*: Stack Overflow posts have been analyzed qualitatively and quantitatively.
    - *Qualitative analysis of posts*: Treude et al. [TBS11b] and Nasehi et al. [NSMB12a] analysed the type of programming posts qualitatively, and classified programming posts into several types (e.g., debugging, how-to questions) In addition, Nasehi et al. [NSMB12a] identified the important common characteristics of a successful programming post (such as using concise code, question context, and step-by-step solutions). Rosen et al. [RS15] investigated the different issues and question types, which face mobile developers.
    - *Quantitative analysis of posts*: Baltes et al. [BDTD18] analyzed the evolution of Stack Overflow posts using string similarity analysis. They found that developers just make

simple changes to posts shortly after their creation. Moreover, source code in posts and text are usually changed together. Wang et al. [WCH18] analyzed the relationship between the time required to get an accepted answer for a question, and the different elements of a post (e.g. users, question, answer). The authors found that answerer reputation has the strongest impact on achieving an accepted answer. They also found that Stack Overflow supports users with higher reputation and who frequently answer questions more than other users. Zhang et al. [ZUR+18] analyzed the quality of answers, and if they propose a high quality solution. The authors found that 31% of the answers could lead to API misuse. For example, some answers in Stack Overflow miss the sequence of calls for an API. Misuse of API can lead to problems like program crash or resource leaks.

2. *Analysing developer behaviour*: Bazelli et al. [BHS13] studied the personality traits of Stack Overflow users using a linguistic inquiry and word count methods. Vasilescu et al. [VFS13] studied the interaction patterns of users between Stack Overflow and Github, to understand the behavior of programmers across the questions asking and answering, and programming activities.

Also researchers analyzed and compared different types of developer communities. Squire [Squ15] conducted an empirical study on comparing the use of Stack Overflow, forums, and mailing lists to support programmers during their development activities. The results show that Stack Overflow has a better support for developers. However, the authors found some groups who moved back from Stack Overflow and preferred mailing lists. Pagano et al. [PM13] conducted an empirical study on software development blogs, where they analyzed the topics in blogs, and their usage from developers. They found that developers use blogs frequently, and specially after software releases. Moreover, they found that developers discuss high-level concepts such as features and domain concepts, while source code related topics are discussed in 15% of their posts.

Analysis and critique

Current empirical studies on developer communities analyze general characteristics of posts, as well as important concepts, properties and categories of programming posts. However, current studies on developer communities do not analyze architecture-relevant posts or architecture knowledge concepts in developer communities. Our studies in Chapters 4 and 5 complement existing analysis studies on developer communities by exploring developer community posts from a new perspective concerning software architecture and focusing on technology-related architecture knowledge.

2.5.2 Using Developer Communities in Software Development

Software engineering researchers proposed approaches to facilitate using developer communities like Stack Overflow during programming activities.

Developer communities contain an enormous amount of information. Therefore, Finding relevant information in developer communities is challenging. Methods to improve finding relevant programming information has been proposed. Gottipati et al. [GLJ11] propose an approach to improve the effectiveness of search engines on technical forums, when searching for a solution to a programming problem. The proposed approach depends on a classification approach, which determines relationships between different elements of a software forum. Experiment results show

an improvement of up to 70% compared to traditional search approaches. Zou et al. [ZYL+15] proposed also an approach to improve search engines on Stack Overflow, when searching for a solution to a programming problem. Their approach depends on re-ranking posts according to the question interrogatives. de Souza et al. [dSCM14] also proposed a search approach, which recommend suitable community posts to a query. The search approach classifies How-to questions (i.e. one of the main types of programming questions [NSMB12a]), and give higher priority to posts with bigger scores. Beyer et al. [BMPP18] proposed a classification approach using machine learning, which categorizes Stack Overflow posts according to the purpose of the question. The authors considered seven categories for programming questions (e.g. Errors, Review). The main goal of the authors is to provide a better categorization for Stack Overflow posts to complement tags, which are mostly technology names.

One of the main attributes of knowledge management is integrating knowledge in products and processes. Thus, approaches have been proposed to integrate programming knowledge in development communities with development environments. Ponzanelli et al. [PBL13a, PBL13b, PBDP+14] created an eclipse plugin called Seahawk to assist developers using developer communities during programming. Seahawk formulates and executes queries automatically based on the context of programming (e.g. current changed source code). Moreover, the tool supports dragging sample code from communities to eclipse in an interactive way. Rahman et al. [RYR14] proposed also an eclipse plugin with a similar approach, where authors combine the results from several popular search engines (e.g. google and Bing) to present the most relevant community posts to the context of programming. The experiments show their approach performs better than using a single search engine.

One well known and commonly used source of knowledge for programmers is API documentation. To improve and complement API documentations, methods to augment API documentation with insights from developer communities are proposed. Treude et al. [TR16] proposed an approach to match the similarity between API documentation and Stack Overflow posts. The approach considers several natural language features like part of speeches. The proposed approach achieves better accuracy compared to summarization techniques and pattern matching techniques. Guerrouj et al. [GBR15] use a summarization technique to provide documentations for classes and methods from sentences in Stack Overflow. To support tasks like code synthesis and code summarization, Yin et al. [YDC+18] proposed automated model creation to learn the relationship between source code and natural language. The approach uses a probabilistic model to capture the correlation between natural language and source code.

Analysis and critique

Software engineering researchers proposed several approaches to capture, share, and support the re-use of programming knowledge in developer communities. The proposed approaches were specially concerned about finding relevant source code in developer communities, which could be reused for certain programming problems. The approaches were additionally integrated with source code editors (e.g. eclipse) to support a recommendation system for programming activities.

Current proposed approaches to re-use the knowledge in developer communities for software engineering activities focus on the implementation phase (i.e. programming) of software, without considering architecture and design phases of software. Our study in Chapter 4 shows that software architects can also benefit from developer communities to support taking technology design decisions. Therefore, we propose in Chapters 6, 7, and 8 approaches to facilitate capturing and finding relevant architecture knowledge in developer communities.

2.6 Summary

Acquiring architecture knowledge for technology design decisions is an interdisciplinary problem, which involves several fields of research. Knowledge management is concerned with general concepts and processes to capture, share, and apply knowledge among individuals, community, and organizations. On the other hand, several areas of software engineering research are related, where each area is concerned with one aspect to support the acquisition of architecture knowledge. Software architecture design processes and methods describe steps and activities to perform architecture design. However, they do not propose or describe any approaches for the acquisition of knowledge. On the other hand, pattern languages provide catalogs of patterns and their relationships to support reasoning about decisions on patterns. Nevertheless, pattern languages do not provide guidance for technology design decisions.

The need to manage architecture knowledge motivated researchers to propose approaches for architecture knowledge management. The proposed architecture knowledge management approaches involve different types of architectural solutions, design decisions (i.e. conceptual and technology solutions), and their relationships. In architecture knowledge management, several models, knowledge sharing and capturing approaches have been proposed. By analyzing the current state of the art, we found that current approaches for architecture knowledge modeling, sharing, and capturing provide minimum support for reasoning on technology design decisions. This is due to the complexity of technology solutions (e.g. they contain different types of features and implement different conceptual solutions). Moreover, the knowledge about technologies is evolving and is distributed among many different sources of knowledge, such as technology specifications, and developer communities. On the other hand, approaches for mining software repositories and developer communities focus on analyzing general characteristics of communities or focus on supporting programming activities.

The analysis of the current state of the art shows that an interdisciplinary solution to acquire architecture knowledge for technology design decisions is needed. Therefore, our proposed solutions in the following chapters consider different areas of research. We first extended in Chapter 3 architecture knowledge models (see Section 2.4.1) with additional concepts to support the reasoning on technology design decisions. Our proposed extension for AK models also consider relationships between technologies and conceptual solutions (e.g. architectural patterns and pattern languages (see Section 2.3)). We then explored and analyzed developer communities in Chapters 4 and 5 to discover technology-related architecture knowledge. Our analysis complements existing empirical studies (see Section 2.5) on developer communities with a new perspective regarding architecture knowledge. Finally, we proposed in Chapters 6, 7, and 8 approaches to facilitate capturing architecture knowledge from developer communities. Our proposed approaches support software architects with relevant knowledge during certain architecture design activities (see Section 2.2).

3

Technology Design Decisions

3.1 Research Question and Contributions 34
3.2 Research Process 35
3.3 Technology Features 38
3.4 Architecturally Significant Technology Aspects 41
3.5 Architecture Knowledge for Technology Design Decision 44
3.6 Evaluation of Concepts 48
3.7 Discussion 50

3.1 Research Question and Contributions

This chapter provides answers for **RQ1** and **RQ2**:
RQ1: How do software engineers conceive software technologies as architectural solutions during the decision making process?

RQ2: How can we model and relate technology decisions with existing architectural knowledge concepts?

By answering RQ1 and RQ2, we support achieving the first goal of the dissertation "Understand technology design decisions" (see Chapter 1). To achieve our first goal and answer RQ1 and RQ2, we conducted an exploratory research study, which consists of a qualitative content analysis, followed by refinement and validation interviews. The study analyzed the different perspectives, which technology vendors, literature and architects have in offering and choosing technology solutions respectively. Moreover, the study integrated our proposed AK concepts for technology design decisions with existing AK concepts in literature. The main **contributions** in this chapter are the following:

- Modeling technology solutions as a set of features, which are offered by the technology vendors, as well as their associated architecturally significant technology aspects (ASTAs), which are considered by architects in taking technology design decision. We determined different categories for technology features and ASTAs.
- An integrated architecture knowledge model, which includes both technology design decision concepts, as well as existing AK concepts from the literature.

Our contributions supported our analysis when answering RQ5 (see Chapter 5) with important AK concepts for taking technology design decisions.

This chapter is structured as follows. In Section 3.2, we describe and explain in details the research process for answering RQ1 and RQ2. Sections 3.3, and 3.4 present our defined concepts of modeling technology solutions as a set of features and aspects. The two sections are followed by Section 3.5, where architecture knowledge concepts for technology decisions are integrated and modeled. In Section 3.6, validation results for the interviews data analysis is presented. Finally, section 3.7 discuss our results.

3.2 Research Process

In order to answer RQ1 and RQ2, we followed the research process shown in Fig. 3.1. The research process is divided into two main phases:

1. *Data Gathering and Hypothesis Definition*: In this phase, the main goal was to collect information about technologies. This would make us understand, what are the primary factors, which make an architect choose or reject a certain technology solution, and what are the scenarios the architect faces during a technology decision. In order to achieve this goal, We followed a qualitative content analysis research method among different technology resources. As a result of this phase, initial hypothesis for the technology decision concepts and models were formulated.

2. *Hypothesis Refinement and Validation*: In this phase, we refined and validated the proposed concepts and models. It was important to align our understanding with what practitioners do in their work. Since the Interviews research method is the best method in discovering human experience [Sei06], we conducted a set of interviews with experts, who are used to take technology decisions frequently. This process helped to improve and validate the proposed model.

In the following sections, we explain both the content analysis and the interview research processes respectively.

3.2.1 Content Analysis

Technology selection guidelines and reasoning are explained in different sources:

1. *Technology Vendors Architecture Guidelines*: Each technology vendor provides guidelines (e.g. [CS10], [PT09]) for designing software systems using their designed products. However, they do not conduct comparisons between products from different vendors, to show their strengths and drawbacks. A list for the analyzed technology specifications and guidelines are available in Appendix A.

2. *Technology Discussion Forums*: This is a rich source for exploring, how technologists choose a technology solution. The discussions show the factors, which drive an architect to choose a certain technology. In addition, comparisons between technologies are usually part of the discussions. For example, on the stackoverflow forum, you might find a topic

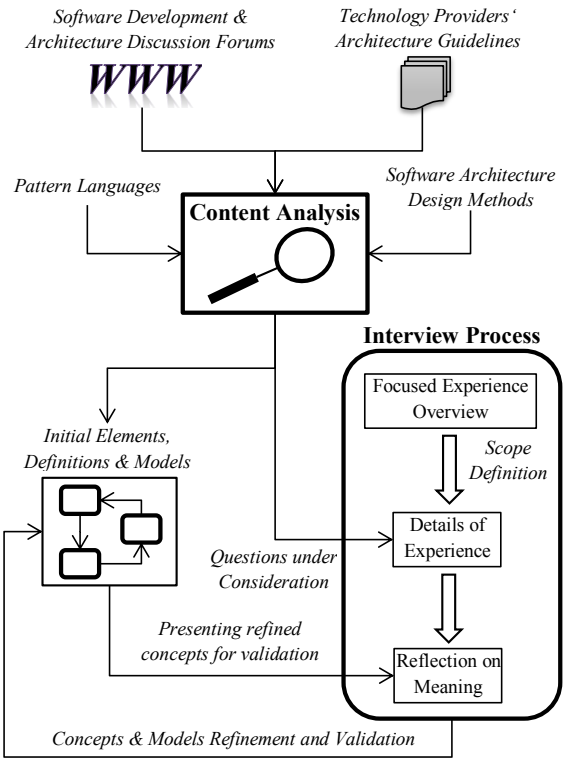


Figure 3.1: Research Process Diagram.

with title "Spring MVC vs JSF". Nevertheless, technology forums lack information about the design reasoning and processes, and their relationship to the architect concerns in taking a technology decision.

3. *Software Architecture Design Methods*: Even though most of the architecture design methods do not provide detailed guidance on taking a technology decision, some architecture design methods (e.g. [BCK12, KC16]) provide guidelines on the different situations, that the architect can face during the selection of technology solutions.

4. *Pattern Languages*: Pattern languages present a catalog of architectural patterns, their descriptions and their relationships with each other (e.g. [BMR+96]). Moreover, each architectural pattern is presented with examples of implementations in technologies. The benefit of analyzing pattern languages is to determine the relationships and interactions between different architectural solutions.

We followed a qualitative content analysis [FvKSJ04] text analysis process through several selected resources from each of the above mentioned categories. Our goal was to have a high-level overview about technology decisions from different sources of knowledge. We selected our data analysis sources based on their popularity, and richness in technology decision knowledge. We followed several guidelines in text analysis, such as coding to refine and categorize the text. The combination of different resources analysis supported us to propose our hypothesis from different perspectives.

Table 3.1: Interview Participants Experience Overview

ID	Exp. (Years)	Technology Background	Role	Industries
1	10	C/C++, Microsoft Products	Technology Consultant	NLP, Performance Critical Systems, E-Commerce
2	13	Microsoft Technologies	Architect	Flight, Communications, Social Media, Reservation, Retail and Education.
3	11	Java / J2EE	Technology Consultant	Billing, Medical-care, E-commerce
4	9	Java / J2EE	Enterprise Architect	Telecom, Costing & Billing, Oil & Mining, Military
5	10	Java / Integration Technologies	Technology Consultant	Communications, Transportation
6	7	Java / J2EE	Technology Consultant	E-Government, Automotive
7	12	Database Systems, Microsoft Products	Enterprise Architect	E-Government, Financial

3.2.2 Interviews

As we were seeking experience in technology architectural decisions, several factors have been considered in choosing the interview participants: 1) Their experience in using and choosing technologies. 2) Their architecture knowledge and design skills. 3) The size of the companies and systems, they are working in or have worked in before. 4) Their interest and motivation to participate in the study. Before choosing the interviewed experts, we identified more than 20 candidate experts through our personal connections. Candidates were evaluated based on the mentioned criteria, to settle on the 7 participating experts. It is interesting to mention that, even though some of the candidate experts work in the role of a software architect in their companies, they do not take technology decisions. In these companies, the architecture decisions are divided between two different roles, the software architects, who design the system conceptually, and the technology consultants who choose the technologies. Therefore, we included in our study experts, who only take technology decisions.

The interviewed experts work, or worked before in software houses or IT service companies, with more than 100,000 employees. All the experts have either a Bachelor or Master degree in computer science or engineering. Due to the fact that the participants live and work in different cities, the interviews have been conducted remotely through telecommunication software.

We followed a three-phase interview process as proposed by Seidman [Sei06]:

1. *Focused Experience Overview*: In this phase, we asked the interviewees to answer several questions to show their experience in software architecture, as well as their technology

experience, projects and domain of work. We made this step one or two days before the first meeting, which supported us preparing the suitable questions which align with the participant's context and experience. Table 3.1 shows a brief summary for the experience overview of the participants.

2. *Details of Experience*: In this phase, we intend to learn from the participant's experience, in order to refine and validate our concepts. The initial hypothesis, that has been concluded from the content analysis phase helped us to direct the questions and discussions. In order to do this, we mapped each concept in the hypothesis with one or more interview questions, which has been customized based on the result from the first phase. During the interview, we were giving the space for the interviewees to explain and express their opinion, and tell us about their experiences, which was the main feedback in this meeting to verify and improve our concepts. The result from this meeting is a set of real practical examples from each participant's experience, which either align, or improve or contradict with our initial hypothesis. In the following sections, we are going to state some of the *Interview Questions (IQ)* that we asked to the experts, as well as their responses[1] and examples. The questions of the interviews are available in Appendix B.

3. *Reflection on the Meaning*: After the first meeting, we were able to refine and verify our initial concepts, through the examples and discussions presented by the participants. However, it was important to validate our interpretation between the experience examples and the proposed concepts. Therefore, in the second meeting, we focused on discussing the proposed hypothesis concepts, and relating it to the mentioned experiences. First, we explained our research goal, and the initial concepts and models, and for each concept, supported by an example from the participant's experience, we asked the interviewee, if this concept align with their understanding and practice. Based on their feedback, we either validated or changed or rejected a certain concept. Sec. 3.6 presents our evaluation results for the proposed concepts, based on the interviewees feedback.

All interviews were recorded and transcripted, to allow further analysis for the discussion. The length for each interview was between 60 and 120 minutes. The difference in duration between the first and the second interview for each participant is between 2 and 7 days.

3.3 Technology Features

One of the main challenges during software architecture design is choosing the right technology solutions (e.g. COTS or frameworks), in order to implement the designed architecture. Even though different technology solutions act as alternative solutions for the same problem, they are different in their capabilities and qualities.

Technology vendors offer their proposed solutions as a set of *Technology Features*. These features are the abstract capabilities, which they claim, that these technologies provide. The vendors usually describe the merits of each feature, and how they could be used to implement a software system. Features could be classified with respect to the capabilities provided into different types. In the following paragraph, we list and define the types of capabilities offered by the features. This list has been derived from our analysis and the interviewees' practical experiences:

[1] Some of the participants' answers have been translated from their native language (Arabic and German) to English.

1. *Development and Configuration Capability*: It provides the ability to develop or configure an implementation for a solution through a development environment, which comprises programming languages and possibly development and testing tools. An example from our analysis is the Microsoft Windows Communication Foundation (WCF) technology feature to develop services for a Service Oriented Architecture (SOA). The services development is done through the C# or VB programming languages, supported by the Microsoft development and testing tools, while, the services' protocols are dynamically defined through a set of XML configurations.

2. *Behavior Capability*: It provides either existing and compiled software components, which implement solutions with a certain quality, or a forecasted behavior for a possible development at certain conditions. For example, several web-based technologies (e.g. JSF, ASP.Net) provide an implemented web process, which embodies HTTP requests handling, and HTML generation functionalities. On the other hand, programming languages' compilers provide different behavioral capabilities, when compiling the source code to object code. For example, the ability of the Java virtual machine compilers to handle multithreading source code among different platforms. Behavior capabilities could be further classified into three capabilities:

    - *Usability Capability*: It provides either an existing user interface functionality, or facilities for developing a user interface. The main target from these capabilities is to facilitate the usability of the developed system. For example, Microsoft Sharepoint provides out of the box user interface features for content and document management. On the other hand, the recent version from HTML 5.0 supports the web designer to use more elegant forms, as well as additional interactive user interface features (e.g. drag and drop) to support an easier front-end development.
    - *Interoperability Capability*: It provides the ability for the technology solution to integrate and communicate with other technologies. The features could be either through an implemented interface, or through supporting a well-known standard or protocol (e.g. SOAP). For example, Java technologies could access Microsoft SQL Server through the JDBC SQL Server Data Access component. On the other hand, Microsoft WCF offers implementations for 9 protocols (e.g. HTTP, TCP/IP, P2P, ...).
    - *Storage Capability*: It provides the ability for the technology to store data, considering the data size, format and processing. For example, Oracle database offers different product editions, the standard edition supports storage with maximum 11 GB, with a single CPU processing, while the enterprise edition has no storage or processing limit.

3. *Operational Capability*: It provides the ability for the technology to monitor and manage the processing of the system during execution. For example, HP Openview products offer features for application status, HW resources, and database system's monitoring.

4. *Commercial Capability*: It is concerned with the price, licenses, and vendor or community support for this technology solution.

Each technology solution encapsulates a tree of inter-related features. Each feature provides a capability from the previously defined capabilities' types[1]. Fig. 3.2 shows an example for a partial technology features' tree, which has been created through our content analysis activity, integrated with the industrial examples mentioned by the interviewed experts. For example, Java Server Faces (JSF) is an open-source technology solution *'Commercial Feature'*, to support developing

[1]We will refer to features, which provide a certain capability type with the capability name, e.g. Development Feature, Behavior Feature

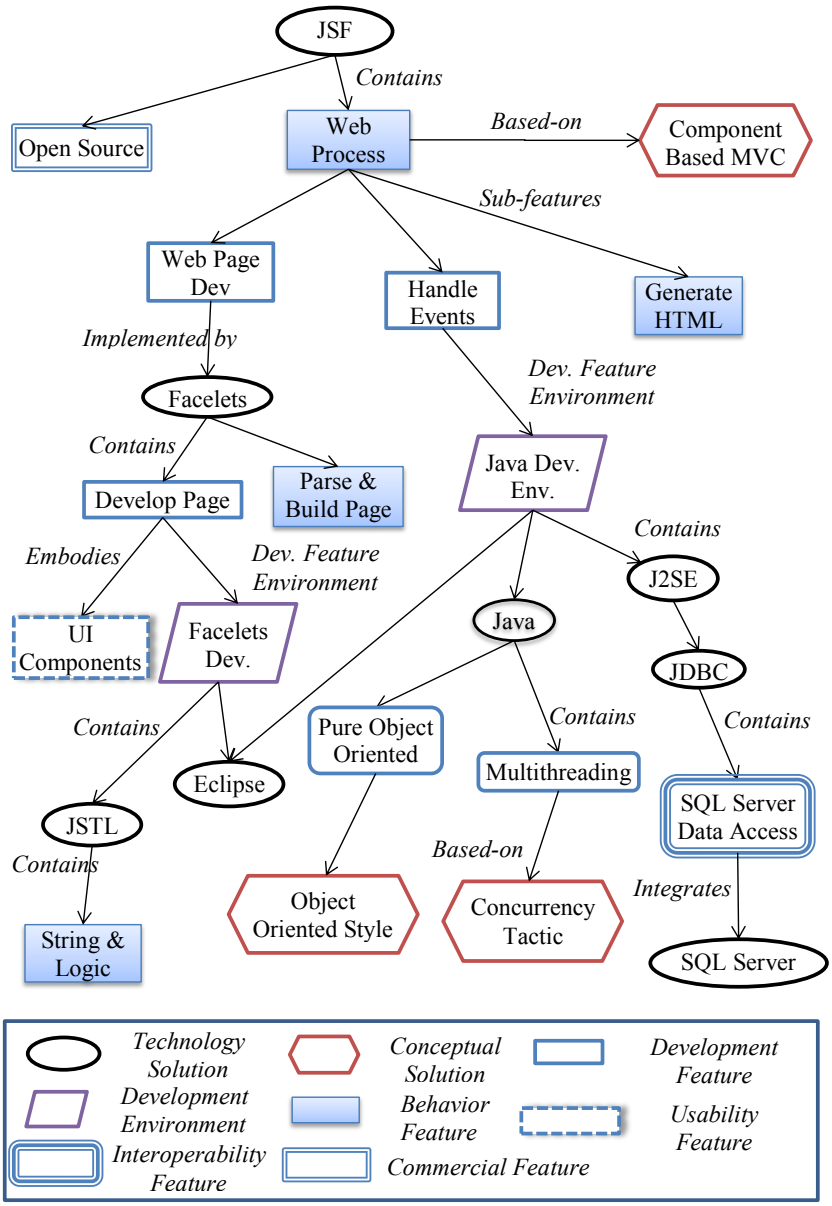


Figure 3.2: An example for a technology features' tree.

a web based application. It contains a web process *'Behavior Feature'*, which is based on the component-based MVC pattern. Several sub-features are associated with the parent web process feature. This includes developing the user interface, handling events *'Development Features'*, and generating HTML *'Behavior Feature'*. Even though, JSF offers development user interface feature, it depends on another technology solution "Facelets" to implement this feature, which consequently offer a development feature, through a development environment. In addition, the Facelets development feature provides a set of user interface components *'Usability Feature'*. On the other hand, JSF event handling development feature is provided through a Java development environment, which incorporates development tools, and the Java programming language with its pure object oriented style. The Java development environment contains several libraries, which embody many additional features. For example, the JDBC library offers an *'Interoperability Feature'* to access Microsoft SQL Server database technology.

Table 3.2: Capabilities' Types and Architectural Concerns Relationships

Capability Type	**Influenced Architectural Concerns**
Development and Configuration Capability	Development Time, Training Time, Maintainability , Testability, Configurability, Evolvability
Behavior Capability	Performance, Reliability, Security, Accuracy, Portability, Reusability
Operational Capability	Manageability, Supportability, Availability
Interoperability Capability	Interoperability, Inter-process communication
Usability Capability	Usability
Commercial Capability	Cost of Ownership, Openness, Development Time, Training Time
Storage Capability	Data Accessability, Scalability

Each of the technology capabilities influence one or more of the architectural concerns, and subsequently each of the offered technology features influences different concerns based on its adopted capability type. Table 3.2 shows the relationship between each of the identified technology capabilities' types and the different architectural concerns [LAH10,Int11]. Understanding the influence of the technology solutions' features on the different concerns supports the architect to evaluate and compare the different technology features, and consequently justify the technology design decision. In Sec. 3.4, we explain how the influence of features on the stakeholders' concerns differentiate the technologies from each other.

3.4 Architecturally Significant Technology Aspects

As explained in the previous section, technology solutions embody many different features, which are designed and implemented within the technology. Consequently, it was interesting for us to ask the participants, "*IQ: To what level does the architect need to know about technology features in order to take the right ADDs?*" Bass et al. [BCK12] listed several important considerations in choosing a technology solution, such as the capabilities of the development tools, the familiarity of the development community with this technology, the possible vendor and community support, the drawbacks of the technology, and the compatibility of this technology with the existing technology stack.

One of the interview participants mentioned that *"The architect doesn't need to know the technology in depth. However, he needs to know the differences between the different technologies, their benefits and drawbacks, regarding performance, vendor support, ..."*, another answered *"He needs to know how technologies work from a high level, considering its learning curve, development effort, usability, ..."*.

Based on our analysis and the interviews' discussions, we define in this section the different types of *Architecturally Significant Technology Aspects (ASTAs)*. They are the principal and distinctive technology solution's characteristics, which distinguish the technology solution from other alternative solutions, and consequently support or influence the architectural design decision. In other words, these aspects qualify such a technology solution to be selected by the architect to satisfy one or more ASRs.

ASTAs could be considered as either *Benefits* or *Drawbacks*. Benefits are the advantages, which the technology solution have over other competitive solutions. On the other hand, drawbacks are either technology features, which are missing in this technology solution, even though they exist in other alternative solutions, or they are well-known existing features' problems, which need to be considered by the architect during the decision making. Both benefits and drawbacks act as different sides of the same coin, such that the same technology solution benefit could be a drawback in another solution. Moreover, both types act as important factors for an architectural design decision.

Each ASTA is associated with a technology feature, and consequently have the same capability type as their related features[1]. One of the interview participants mentioned *"Even though Java provides an important feature for code portability among different platforms, this is a significant drawback for our development, which seeks native components development. This makes us always favor C as our development technology. Nevertheless, it lacks such a platform independent feature"*. Furthermore, we believe that both benefits and drawbacks are relative notions, which could be solely determined through a comparison with other competetive technologies. Such a comparison is not usually provided by the technologies' vendors. However, they are part of the software community discussions and experiences.

Due to the fact that features are commonly described ideally by the technology vendors, without mentioning their drawbacks, several discussions on the technology forums try to share their experiences with either problems, which they faced in using these features, or missing capabilities, which were expected to be provided. Associated with these discussions are side-by-side comparisons between features from different technologies, which show the benefits and drawbacks of each feature in comparison to the other technology features. We call these types of aspects *Feature-Based ASTAs*.

One of the interview participants mentioned the following: *"In order to choose a web-based framework, we depended on a comparison between Spring MVC and other frameworks. We prefered to use the Spring MVC framework over other frameworks, because it's supported with better documentation, which make it easier to develop and learn."* In this situation, the interviewee took the design decison based on a development feature-based benefit, which is the "better development documentation".

On the other hand, as we mentioned in Sec. 3.3, each feature influences different types of architectural concerns. This influence could be measured differently depending on the type of feature's

[1] We will refer to ASTAs, which assess a certain capability type with the capability name, e.g. Development ASTA, Behavior ASTA, ...

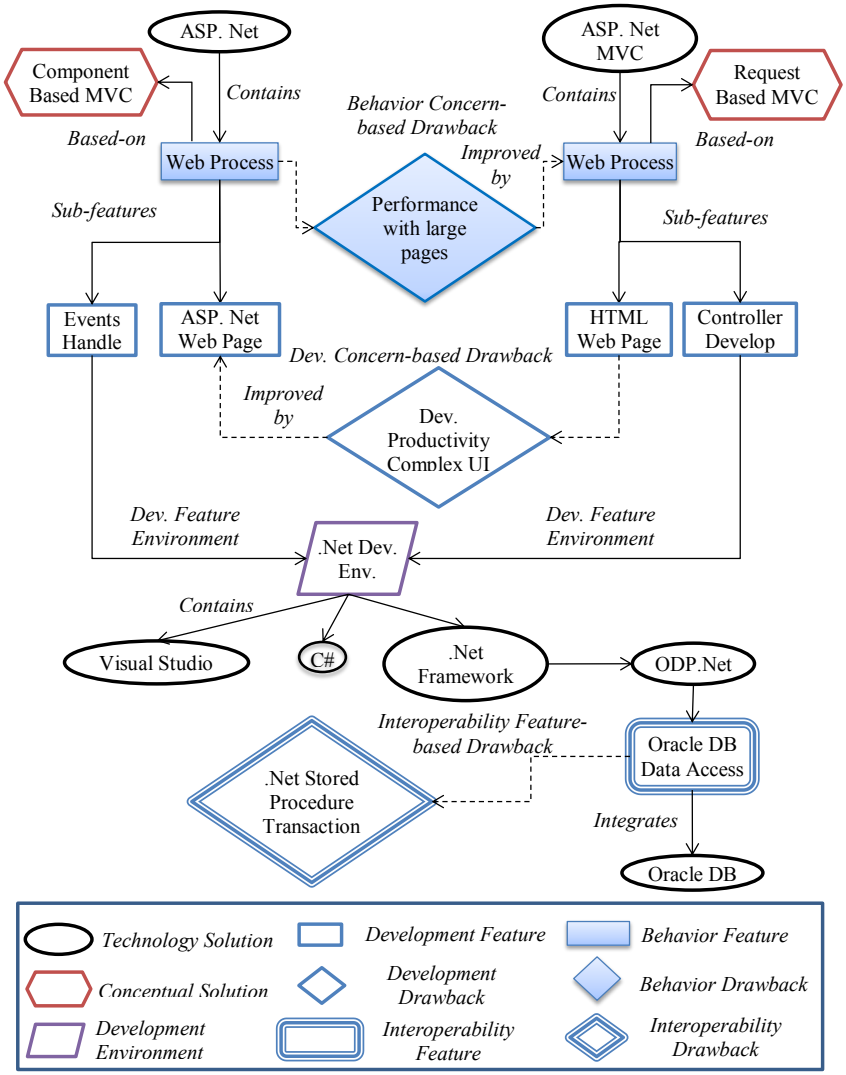


Figure 3.3: A subset from a technology features' tree with both feature and concern based drawbacks (ASTA).

capability and the assessed concern. For example, benchmarking can be a suitable method for comparing the performance of the behavioral features for multiple technology solutions. In addition, development features could be measured through development productivity experiments (e.g. [Phi99]). As a result of these measurements, we could recognize, which technology solution features are better than the others regarding the measured concerns. In other words, the concern-based benefits and drawbacks or *Concern-Based ASTAs*.

An interview participant described an experience about a certain database management system *"After 1 year, the amount of data in the database reached more than 20 billion records. After several testing, we discovered that the performance of data processing operations is degrading exponentially with the increase in the amount of data, this technology drawback derived us to take an architectural decision to replace such a database management system"*.

Fig. 3.3 shows two web interface technology solutions, ASP.Net and ASP.Net MVC. Both technologies are based on the MVC architectural pattern. However, they are based on different pattern variations. This difference triggers a distinction in their development and behavior features. An interviewee mentioned *"According to our experience, ASP.Net provides a feature for faster and easier development environment than the ASP.Net MVC, due to the availability of many reusable UI components. However, ASP.Net is slower than ASP .Net MVC in handling thousands of users, due to the fact, that Component-based MVC need to store the view state of each UI control."* By analysing this statement, we could identify the following ASTAs: A) ASP.Net faster and easier development than ASP.Net MVC is a *'Development Concern-Based Benefit'*, B) The availability of many reusable UI components in ASP.Net is a *'Development Feature-Based Benefit'*, C) ASP.Net is slower than ASP.Net MVC in handling thousands of users is a *'Behavior Concern-Based Drawback'*, and D) Component-based MVC need to store the view state is a *'Behavior Feature-Based Drawback'*.

From this experience example, we can conclude that both types of ASTAs are strongly related, such that an identified feature-based drawback would impact the feature negatively, which might lead to a concern-based drawback. Similarly, a feature-based benefit would impact the technology feature positively, which possibly lead to a concern-based benefit.

3.5 Architecture Knowledge for Technology Design Decision

In this section, we consolidate and model the technology features and ASTAs' concepts, which are presented in the previous sections. In addition, we integrate the proposed technology concepts with the reusable design decision concepts proposed by Zimmermann et al. [ZKL+09]. Finally, we appended additional decision making elements to formulate our proposed architecture knowledge (AK) model. The main elements of the proposed architectural knowledge model are presented in the first sub-section 3.5.1. In sub-section 3.5.2, we further discuss and classify architectural solutions according to their ability to interact between each other in a decision making process. The proposed model and classifications support architects to make technology design decision.

3.5.1 Elements of Architectural Knowledge

Fig. 3.4 shows the meta-model for the proposed architecture knowledge. At the core of the model are the design issues, which are the architecture design problems facing the architect. Each design issue is associated with a set of alternative architectural solutions [ZKL+09]. An architectural

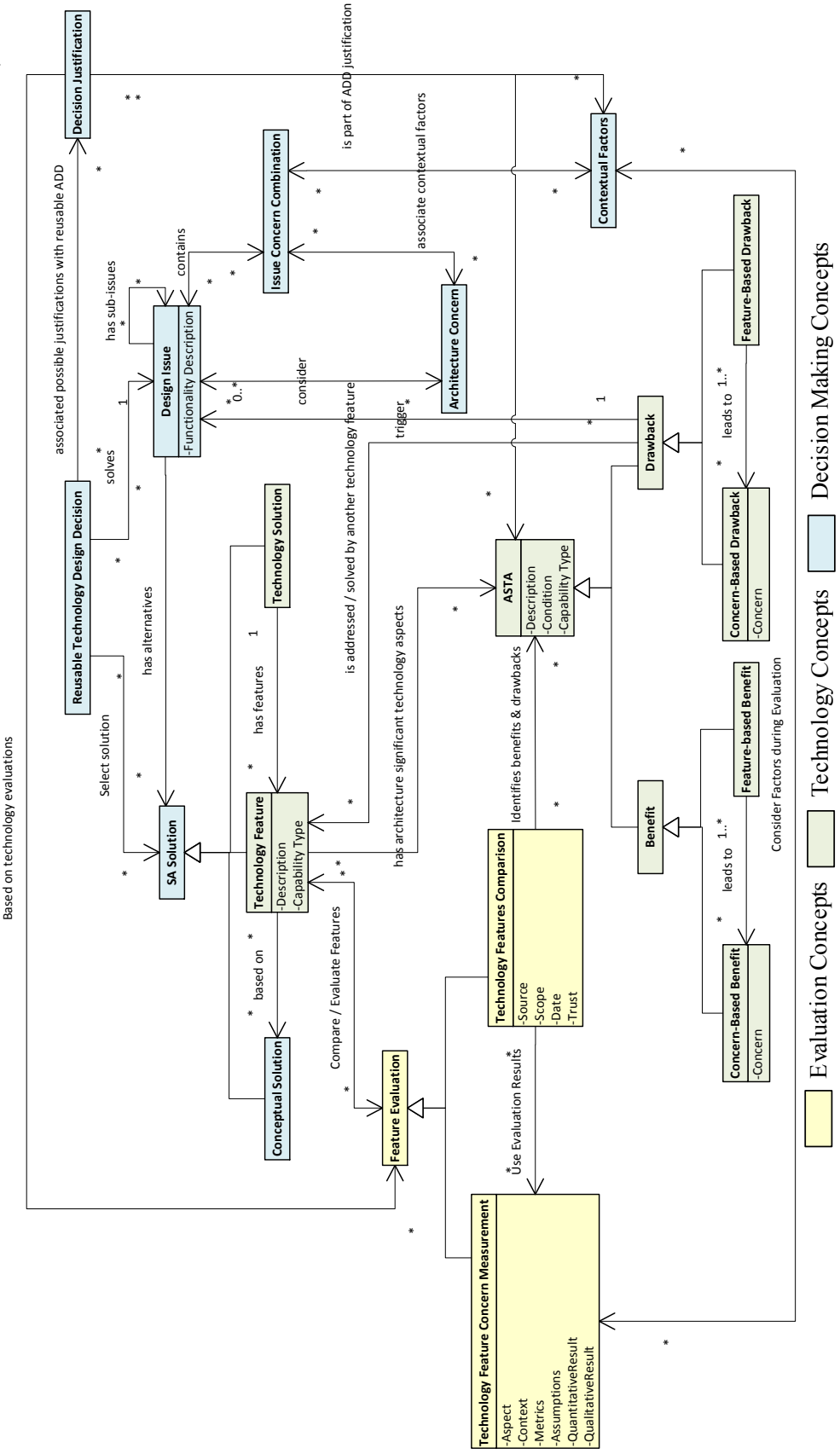


Figure 3.4: Software Architecture Knowledge (AK) Metamodel.

solution could be either a conceptual solution (e.g. architectural pattern or tactic), or a technology solution, or a feature within a technology solution. As explained in the previous sections, each technology feature may have certain ASTAs, which are identified through comparisons between different technology features. ASTAs could be either benefits or drawbacks, feature-based or concern-based. Moreover, the identified ASTAs are part of the technology decision justification.

A feature's drawbacks are usually sources of additional design issues, which the architect need to overcome before selecting the technology solution. Overcoming solutions are different, however, one of their options is using other technologies, which contain features that address this problem. For example, one of the interview participants mentioned *"JSF framework has drawbacks, which impact the usability quality attribute. Two options were proposed, either to use an additional framework (e.g. Primefaces) to mend this problem, or hire a web designer to fix the problem manually"*.

During our interviews, we asked all the participants the following question: *IQ: What are the steps you take to choose between a list of possible technology solutions?* It was not surprising that each participant described a different reasoning process than the others. However, all participants shared a common set of elements, which they consider in selecting the technology solution, even though, they are used in a different order. Therefore, the proposed AK model contains and relates the different elements, which are necessary to assist the architect in taking a technology design decision, such that it can align with the different design reasoning methods.

Even though, each of the interview participants described different steps in choosing a technology solution, we can still group their description into two main reasoning types [TvV12, Tan11]:

1. *Deductive, problem-driven*: In this process, the architect starts by analyzing the design issue, and its associated architectural concerns. For example, in choosing between different middleware technologies, interoperability, performance, security and development time should be considered. For each of these concerns, several factors are associated; the technology at the client and server, the size and structure of the transfered data, the network between the two sides, and the development team skills respectively. Based on these factors, the architect can start evaluating the different technology features (e.g. [GLB03]). Alternatively, the architect could assess the feature's quality through evaluating the conceptual solution (e.g. evaluation for architectural patterns [BR10]), which this feature implements. By the end of the process, the architect needs to make trade-offs among the different concerns (e.g. using [ANGB+05]).

2. *Inductive, solution-driven*: In this process, the architect starts by checking other experiences, which have similar situations. In other words, design decisions which have been taken in different projects but with similar circumstances, and based on matching both conditions, the architect chooses the suitable technology solution. We believe that technology decisions that are justified based-on the architectural concerns and factors are reusable, such that similar architectural concerns and factors could be repeated among different projects. On the other hand, technology decisions that are justified based on business or social aspects cannot be reused within the context of software design decision (maybe arguably reused in a business context).

Even though our main research goal is not to drive a design process, it is important to understand the different reasoning methods, which the architect uses, in order to identify the important architecture knowledge elements and their relationships.

3.5.2 Solutions' Interactions within an Architectural Knowledge

During an architectural design process, software architects select a sequence of different architectural solutions. Each architectural solution has a different impact on the quality attribute of a software system, as well as a different impact on the resulting structure of the system components. Based on our content analysis, we classified architectural solutions into two main types:

1. **Triggering Architectural Solutions**: They are the type of solutions that have the ability to trigger new architectural design issues, such that in order to complete the architectural design of these solutions, new architectural design issues must be addressed. In this group belong architectural patterns, and technology solutions.
2. **Elementary Architectural Solutions**: These are architectural solutions that do not trigger new architectural design issues. They are either as architectural unit operations (e.g. Component Decomposition) or solutions recommendations for a subsequent detailed design (e.g. Design Patterns [GHJV94]).

Architectural tactics [BCK03] and technology features (see Section 3.3) have a different nature as other solutions, such that it is hard to classify them all in a single group.

Tactics could be divided among the two groups, into elementary and triggering tactics. Elementary tactics are tactics that do not trigger new architectural design issues. For example, to improve the performance of a well-known process (e.g. Products sorting), selecting or changing the algorithm usually would not produce new architectural design issues, however, it can produce algorithmic or implementation issues. On the other hand, triggering tactics require more architectural design issues to be addressed in order to realize the design of the tactic. For example, improving the performance through caching, this would require to answer other design questions such as: which and where to cach the data? and how to synchronize the cached data?

Technology features could also be divided among the two groups, into elementary and triggering technology features. The classification depends on the influence of a technology feature on other features. Triggering technology features are main features of a technology solution, which consists of other features. In many cases, it is obligatory to use a triggering technology feature, if an architect decided on this technology solution. On the other hand, elementary technology features are smaller features, which are optional to select. They have a smaller influence on other features. In Fig. the "web process" technology feature of the JSF technology is one of the main features in JSF, which could be classified as a triggering technology features, because many other features depend on it. On the other hand, the selection of the JDBC SQL Server Data Access technology feature is an elementary technology feature, which do not influence other features in JSF.

Architectural design issues vary in their importance, types, scope and position within the reasoning process. Zimmerman et. al Zimmermann et al. [ZKL+09, Zim11] classified design issues based on their abstraction level. In order to support the architect in understanding when design issues occur within the reasoning process. We propose a classification for design issues, based on their occurrence within the design reasoning process and their relationship to the architectural solutions.

1. **Root Design Issues**: They are design issues which are stimulated independently from previously selected architectural solutions. Enterprise or principal high level design issues (e.g. deciding the high level architectural style of the system or the main implementation technology) are popular examples that belong to this group.

2. **Solutions-Triggered Design Issues**: They are design issues that must be triggered based on a stimulation from a previously selected architectural solution. We further classified these issues based on their relationship to the architectural solutions into the following groups:
   (a) **Solution-Specific Design Issues**: They can only be triggered as a result of selecting a specific solution. They can't be triggered by any other solutions.
   (b) **Joined Design Issues**: A common design issue which can be triggered by different solutions.
   (c) **Integration Design Issues**: This is a type of design issue which is conditionally triggered as a result of selecting two or more architectural solutions. It represents the integration design problem between the different architectural solutions.

Fig. 3.5 shows an example of a subset of issues and solutions that are triggered as a result of selecting the Layer architectural style and the MVC architectural pattern. Both solutions were triggered as a result of two root design issues, independently from any previous solution selected. However, they are influenced by several decision factors (e.g. requirements, team structure, ...). In order to realize the design of both triggering solutions, several design issues have to be addressed. For example, to define the Layer structure, an abstraction paradigm (e.g. distance from hardware or complexity) must be defined, this decision can depend on several factors (e.g. system domain). Similarly, in order to design the relationship between the Model and Views/Controllers within the MVC pattern, a 'change propagation mechanism' (e.g. using a Publish-Subscribe pattern) must be selected. Both of these issues are examples of solution specific design issues. On the other hand, designing the domain components of the system is required to be addressed for both solutions, however, for two different purposes. Firstly, to define how objects are communicated between layers, and secondly to provide a separation of concerns between the Model and View components. Finally, the introduction of both the Layer and MVC solutions together triggers an issue, whose purpose is the integration of both solutions components.

3.6 Evaluation of Concepts

In this section, we qualitatively evaluate the agreement among interview participants on the technology design decision concepts in Sections 3.3 and 3.4.

3.6.1 Interview Responses Analysis Results and Observations

Table 3.3 shows the data analysis results for the interviewees' responses. In order to accurately evaluate the feedback of the participants for each of the explained concepts in the previous sections. We designed several levels of responses:

1. *Concept Contribution*: The participant mentioned based on his experience a new concept or an improvement to a concept which was not originally part of the content analysis derived hypothesis.
2. *Concept Supported*: The participant supported the hypothesis concept with additional examples from his experience.
3. *Concept Accepted*: The participant accepted the proposed concept. However, she does not have an example from her experience to support it.

Table 3.3: Interview Data Analysis Results: ++: *Concept Contribution,* ✔*: Concept Supported, Y: Concept Accepted, N: Concept in Doubt, –: No Answer Provided*

Concept vs. Participant	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**%**
Technologies as Features & ASTAs (Sec. 3.3 & 3.4)	✔	✔	✔	✔	✔	✔	✔	100
Development & Configuration Feature or ASTA (Sec. 3.3 & 3.4)	Y	N	✔	++	✔	✔	Y	86
Behavior Feature or ASTA (Sec. 3.3 & 3.4)	✔	Y	✔	✔	✔	✔	✔	100
Usability Feature or ASTA (Sec. 3.3 & 3.4)	Y	Y	Y	✔	Y	✔	✔	100
Interoperability Feature or ASTA (Sec. 3.3 & 3.4)	–	++	Y	Y	++	–	✔	71
Storage Feature or ASTA (Sec. 3.3 & 3.4)	✔	Y	Y	Y	Y	–	✔	86
Operational Feature or ASTA (Sec. 3.3 & 3.4)	–	–	–	–	++	Y	✔	43
Commercial Feature or ASTA (Sec. 3.3 & 3.4)	Y	Y	++	Y	Y	–	✔	86
Decision Making Factors (Sec. 3.5)	Y	✔	✔	Y	✔	–	Y	86
Decision Making Concerns (Sec. 3.5)	✔	Y	✔	✔	✔	–	✔	86
Decision Making Evaluation Report (Sec. 3.5)	Y	–	✔	–	–	✔	✔	57
Decision Making Deductive Problem Driven (Sec. 3.5)	Y	✔	✔	✔	Y	Y	✔	100
Decision Making Inductive Solution Driven (Sec. 3.5)	Y	–	–	–	✔	✔	✔	57
Drawbacks ASTA (Sec. 3.4 & 3.5)	✔	Y	++	Y	++	✔	✔	100

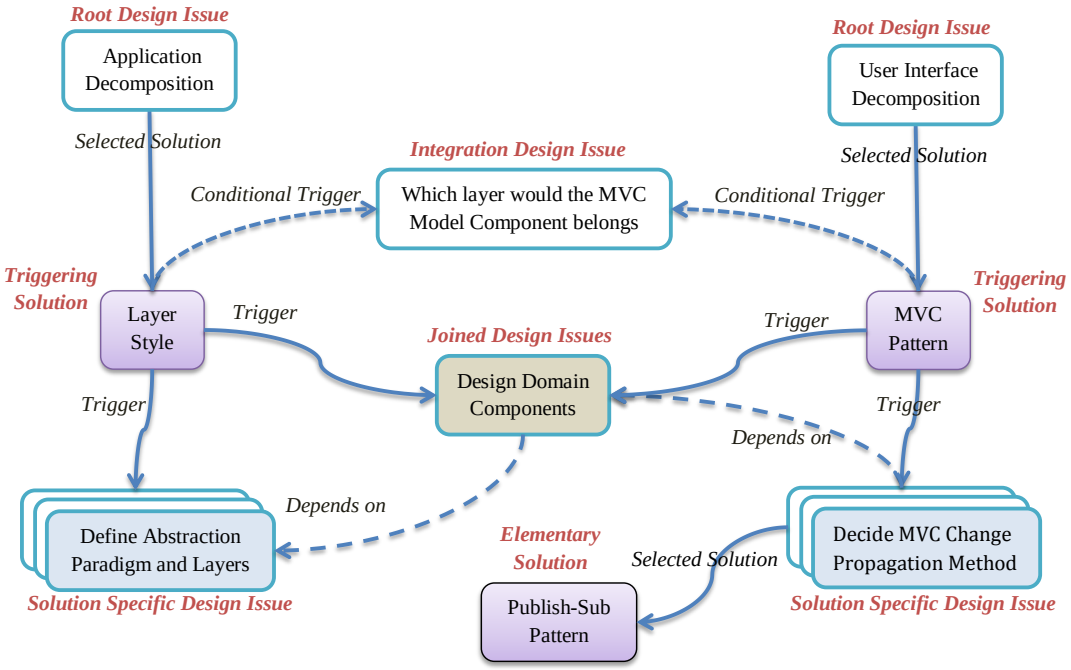


Figure 3.5: An example showing the interaction of objects based on the proposed architectural knowledge domain model

4. *Concept in Doubt*: The participant indicated that the concept is unclear to her, or it does not align with her experience.

The concepts which were characterized as unclear by the majority of the participants have been removed.

All the concepts have been experienced by the participants during their decision making experience. We can observe that the drawback ASTAs concept (Sec. 3.4 & 3.5) and the interoperability features and ASTAs (Sec. 3.3 & 3.5) were the mostly contributed concepts due to their importance to the participants. However, some of the contributed concepts have not been evaluated by all participants, due to the fact, that the interviews have been conducted incrementally.

3.7 Discussion

3.7.1 Interpretation of Results

Two main important and useful outcomes of this study:

- *Knowledge concepts*: The study specified the most important AK concepts (i.e. technology features and ASTAs) for technology design decisions based on empirical evidence from interviews and literature. Moreover, the study showed that ASTA concepts are paramount for architects to choose among existing technology solutions and features.

- *Knowledge sources*: The study explored different sources of AK, which contain technology related architectural information. Our content analysis for technology specification documents and our interviews with practitioners gave the first overview on possible AK concepts, which exist in each AK source. The study showed that technology specification documents are rich with a detailed description about technology features. However, they do not compare or relate different technology solutions with each other. On the other hand, developer communities have the advantage to describe the benefits and drawbacks of different technology solutions (i.e. ASTAs).

3.7.2 Implications of Results on further Research

Conducting this study to answer RQ1 and RQ2 was quite useful to determine further goals and research questions in this thesis. Our new understanding for the distribution and complexity of technology-related architectural knowledge helped us to realize that effective approaches for knowledge extraction are needed to capture technology-related architectural knowledge. This is especially needed because of the fast production and appearance of new technologies, which software architects need to learn and cope with. Moreover, the distribution of knowledge among technology specification documents and many of the existing developer community web sites makes it challenging to find relevant architectural information. In chapters 4 and 5, we explore developer communities for architectural knowledge. During our analysis of developer community web pages, we used the concepts in our proposed architecture knowledge model in Section 3.5.1 to capture their concrete textual representation in community web pages. Finally, we propose in Chapters 6, 7 and 8 approaches to facilitate finding relevant architectural information in developer communities.

Part II

Analysis of Architecture Knowledge in Developer Communities

4

Architecture-relevant Posts in Developer Communities

4.1 Research Questions and Contributions 53
4.2 Research Process 54
4.3 Types and Variations of Architecture-relevant Posts 58
4.4 Practitioner Evaluation 71
4.5 Discussion 74

4.1 Research Questions and Contributions

This chapter provides answers for **RQ3** and **RQ4**:
RQ3: What are the types of architecture-relevant posts (ARPs) in developer communities?

RQ4: Which types of ARPs in developer communities do practitioners consider architecture-relevant?

By answering RQ3 and RQ4, we support achieving the second goal of the dissertation "Explore developer communities for architecture knowledge" (see Chapter 1). To answer RQ3 and RQ4, we decided on selecting Stack Overflow (the biggest developer community website) as our sample developer community. We selected a sample of Stack Overflow posts, and performed qualitative content analysis to classify and characterize Architecture Relevant Posts (ARPs). In addition, we evaluated our classifications through a feedback evaluation with practitioners to answer RQ4. In summary, we present the following contributions:

- We define architecture-relevant posts and classify concrete sub-types for ARPs.
- We evaluate the agreement of practitioners with our classification of ARPs.
- We provide the first corpus of evaluated architecture relevant posts from a developer community.

Our contributions support achieving further research steps. The corpus of architecture-relevant posts has been used to provide the sample for a qualitative content analysis to develop an ontology for the architecture knowledge concepts in developer communities. This answers RQ5 (see Chapter 5). In addition, the corpus has been used in Chapter 7 to train machine learning classification

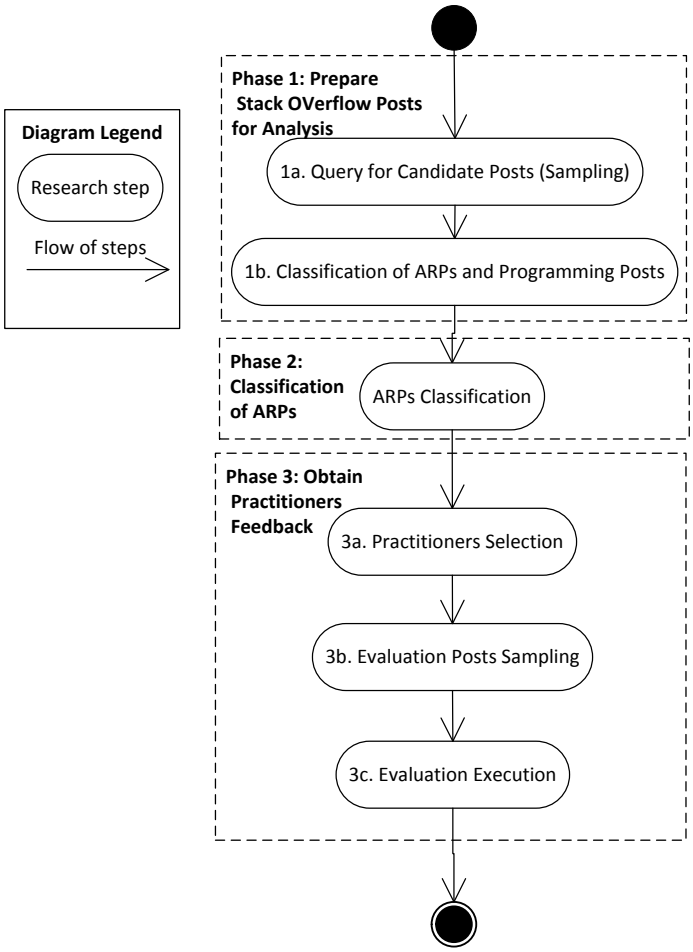


Figure 4.1: Research Process Diagram.

algorithms for posts classification. Finally, the identified types of ARPs support providing suitable architecture knowledge for design activities. Using this aspect, we proposed our enhanced search approach in Chapter 8.

The chapter is organized as follows: In section 4.2, we describe our research process. In section 4.3, we define and present the different types and variations of architecture-relevant posts (ARPs). This answers RQ3. In section 4.4, we present our evaluation results for the different types of ARPs. This answers RQ4. Finally, section 5.4 discuss our findings.

4.2 Research Process

We followed the research process depicted in Fig. 4.1, which consists of three phases.

4.2.1 Phase 1: Prepare Stack Overflow Posts for Analysis

4.2.1.1 Phase 1a: Query for Candidate Posts (Sampling)

Stack Overflow posts could be classified into different topics (e.g. [BTH14]) Some of these topics are architecture related (e.g., posts about web development and database platforms). On the other hand, posts about coding styles and string manipulations are examples for topics which are programming related. As our main objective is to explore architecture relevant Stack Overflow posts with technology-related AK, we selected the "Middleware" topic as our sample topic for two reasons:

1. Middleware is an important topic in the software architecture field, for example, many architecture patterns (e.g. [BHS07a, VKZ04]) address interoperability issues.
2. The amount of middleware posts in Stack Overflow is manageable for our exploratory study (2,561 posts that meet our selection criteria, see next paragraph) are manageable for our study. In contrast, we counted more than 12,000 post about web development which meet our selection criteria mentioned in the next paragraph (except criterion 1 which is about middleware).

In order to gather candidate architecture-relevant middleware posts, we applied the following selection criteria:

1. Posts had to be concerned with at least one of 52 different middleware technologies (e.g. RabbitMQ, WCF). We queried post title, questions and tags for middleware technologies. The 52 selected technologies are the most popular ones in the middleware domain. We identified these technologies through a review of existing technologies on Wikipedia[1]. A complete list of technology names is available in Appendix A.
2. We excluded posts with no answer.
3. We considered posts with a question score higher than or equal 7 (similar heuristics have been applied in other related work [NSMB12a]). This was to ensure the quality of the selected posts.
4. We excluded posts which include blocks of source code in the question because most of those posts discuss programming problems [NSMB12a]. The gathering process has been done using a set of SQL queries through the stack exchange explorer [2] and resulted in 2,561 posts. The SQL query is available in Appendix A. A complete list of all Stack Overflow posts is available online[3] and in the companion CD.

[1] https://en.wikipedia.org/wiki/Category:Message-oriented_middleware
https://en.wikipedia.org/wiki/Category:Enterprise_application_integration
[2] https://data.stackexchange.com/stackoverflow/query/new
[3] https://swk-www.informatik.uni-hamburg.de/~soliman/Dissertation.zip

4.2.1.2 Phase 1b: Initial Manual Classification of ARPs and Programming Posts

Two experienced researchers[1] classified the same sample posts manually to differentiate architecture-relevant posts from programming posts. Furthermore, we excluded posts which were included after the query-based search in Phase 1a but turned out to be not about middleware. We conducted reliability tests to ensure that the classifications of the two researchers were consistent. For that reason, we conducted three iterations and checked for inter-coder reliability (% of agreements) and calculated the kappa coefficient [Coh60] to determine whether agreement between researchers was beyond chance. In each iteration, 100 posts were randomly selected and classified by two researchers, different types of posts and examples were gathered, and disagreements were discussed. By the end of these iterations, we defined architecture-relevant posts, together with a set of classified post examples, according to the prototype theory of definition [May14]. Based on this definition, we classified the rest of our sample as ARPs and programming posts. This step resulted in 858 ARPs, and 1,653 programming posts, while 50 posts have been excluded for being out of middleware scope. We list the classified Stack Overflow ARPs in Appendix A.

4.2.2 Phase 2: Classification of ARPs

In order to answer RQ3 and to classify the ARPs into types based on the concerns that they address, we followed a summarizing qualitative content analysis method [May14], where a short summary about each post is written and assigned to a category. We applied this method to the gathered 858 candidate ARPs from phase 1b. During this process 30 categories of posts emerged. By the end, we refined these categories and combined them into higher level categories. The identified categories are presented in Section 4.3 as our ARP types.

4.2.3 Phase 3: Obtain Feedback from Practitioners

In order to evaluate our classification of ARPs and ARP types identified in phase 2, and to understand the perspective of practitioners regarding the different ARP types, we involved practitioners to classify a set of posts as ARPs or programming posts.

4.2.3.1 Phase 3a: Practitioners Selection

To identify participants, we considered the overall software development experience, as well as the software architecture experience. In addition, we considered the variation of technology experiences and backgrounds. All of the participants are familiar with Stack Overflow and use it as part of their work. All participants work or worked in multinational companies with more than 100,000 employees. We identified participants through contacting them directly. 11 experienced practitioners participated in the evaluation of posts. In Table 4.1, we present background information about the participants.

[1] The first reseracher is the author of the dissertation. The second researcher is Matthias Galster from the university of Canterbury

Table 4.1: Background of participants

ID	IT Exp. (Years)	Software Arch. Exp. (Years)	Technology Background	Industries	Role
1	11	4	C/C++, Microsoft Technologies	NLP, E-Commerce	Technology Consultant
2	10	4	Java / J2EE	Telecom, Billing, Defense	Enterprise Architect
3	11	5	C++, WebLogic, Spring, J2EE	Telecom	Technology Consultant
4	14	7	Microsoft Technologies	eGov, Financial, Healthcare	Solution Architect
5	17	8	Microsoft Technologies, BizTalk, Azure	Healthcare, Manufacturing	Program Manager
6	10	4	Java / J2EE	Healthcare, Retail, Billing	Technology Consultant
7	11	5	J2EE, Weblogic, Unix, RabbitMQ	Telecom	System Engineer
8	12	4	Microsoft Technologies, NodeJS	Transportation	Technology Consultant
9	30	12	VME Mainframe, Siebel, J2EE, Unix, WebSphere	Defense, Insurance, eGov, Manufacturing	Solution Architect
10	13	4	Microsoft Technologies, Biztalk, JBoss, Node JS, C++, Python	Telecom	Solution Architect
11	6	1	Microsoft Technologies	Retail	Developer

4.2.3.2 Phase 3b: Evaluation Posts Sampling

In order to make the sample of posts given to practitioners representative to our overall sample of 2561 posts, we conducted stratified random sampling [RR08] among the ARP types (which are presented in Section 4.3), as well as the programming posts. This way of sampling guaranteed that all types of posts were included in the evaluation sample proportional to their frequency within the analysis sample.

4.2.3.3 Phase 3c: Evaluation Execution

In addition to the data shown in Table 4.1, we asked our participants about their definition of software architecture, in order to ensure their understanding of software architecture. Each participant received an "evaluation sample" of posts. The total number of posts used in the evaluation was 1,173 posts (The numbers of posts given to each participants are listed in Table 4.26 in Section 4.4). It took each participant between 3 and 4 hours to classify the posts between ARPs and programming posts. The analysis of the feedback of the participants helped us to answer RQ4, which is presented in Section 4.4.

4.3 Types and Variations of Architecture-relevant Posts

In this section, we present the types of architecture-relevant posts (ARPs), which we discovered in our sample of Stack Overflow posts. This answers RQ3.

4.3.1 Types of Architecture-Relevant Posts

Based on our analysis (see Section 4.2), we identified three main types of Stack Overflow posts according to the concerns mentioned in the questions of a post. We named them *Programming Posts, Architecture-Relevant Posts (ARPs)*, and *Cross Architecture/Programming Posts (CAPPs)*.

Programming Posts are posts with questions related to performing a programming activity. The answers to questions in programming posts often include source code fragments, step-by-step guidelines on how to code a feature, or lower level technology details. This type of post constitutes the majority of Stack Overflow posts. Nasehi et al. [NSMB12a] classified programming posts into four types:

- *Debug/Corrective*: Concerned with problems and errors in source code.
- *Need to Know*: Asking about programming features such as classes and methods.
- *How to Do it*: Asking about how to implement a certain functionality or scenario.
- *Seeking different solution*: The user has a working code and searching for alternative source code solutions.

Since these types of posts are not our focus, we do not discuss their details. The reader may refer to existing empirical studies on programming posts (e.g. [NSMB12a, TBS11a]).

Architecture-Relevant Posts (ARPs) are posts with questions related to performing an architecture design activity. In addition, the questions in an ARP sometimes consider quality attributes and contextual factors.The answers to these questions involve experience and knowledge about technology solutions, their differences and capabilities. This type of posts is the focus of this paper. Therefore, in order to better understand ARPs, we classified ARPs based on two dimensions: (1) the purpose of the question, and (2) the solution type of the question. Along the purpose dimension, ARPs could be classified into the following sub-types:

1. *Solution Identification*: concerned with searching for suitable technology solutions, which have certain characteristics (such as technology features, quality attribute evaluation); address a design problem or context.
2. *Solution Evaluation*: concerned with assessing one or more proposed technology solutions. The evaluation of solutions could be done individually or through a comparison between different alternative solutions. In addition, several concepts are considered during evaluation, such as technology features, benefits and drawbacks, suitable use cases, and quality attributes.
3. *Multi-purpose*: this type of ARP comprise both types of posts; solution evaluation and identification. Several questions are asked within a single post.

Note that there might be other purposes for asking questions in ARPs. We only identified the purposes based on our sample.

On the solution type dimension, ARPs could be classified as follows:

1. *Technology Bundle*: consider the whole technology as a single architecture solution, without focusing on specific features within the technology. Technology bundle solutions are usually referred to using technology names (e.g. WCF).
2. *Technology Feature*: focus on specific features of a technology (such as deployment, authentication). For example, Gorton et al. [GKN15] provide a taxonomy for features in database management systems.
3. *Architecture Configuration*: concerned with the components and connectors design configuration [MT00]. The types of components and connectors could either belong to a technology feature (see above) or bundle (see above) or a conceptual solution (such as an architectural pattern or tactic). Note that conceptual solutions are not a category of ARP's above since we focus on technology solutions. Also, we did not find many pure conceptual posts.
4. *Combined Solution*: concerned with different solutions types. The post may consider two or more of the aforementioned solution types.

Similar to the purpose classification dimension, there could be additional solution types. We listed types identified in our sample.

By combining the purpose and solution type dimensions, we can specify types of ARPs. For example, an ARP which is about evaluating a solution (purpose dimension) and discusses a technology bundle (solution type dimension) is a post, which is concerned with evaluating a technology bundle solution (e.g. a framework), we call it "*Technology evaluation*" ARP. On the other hand, an ARP about solution identification (purpose dimension) and discusses an architecture configuration

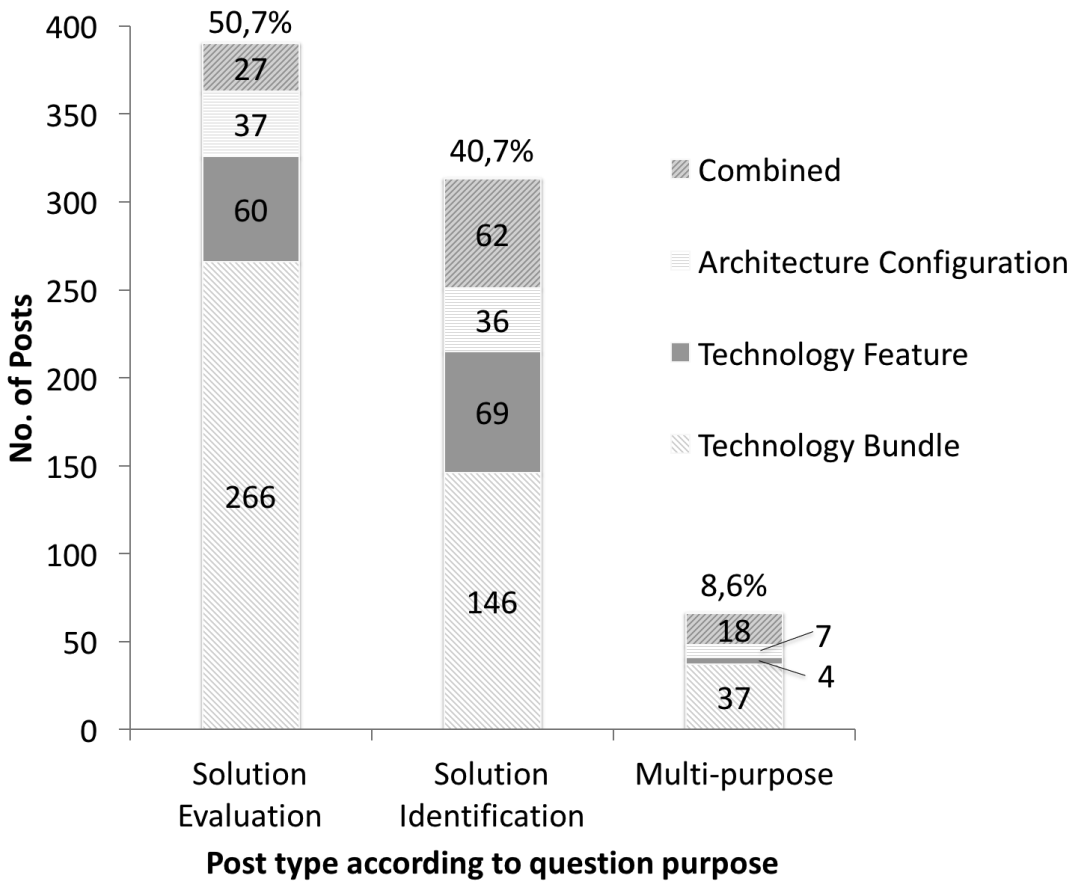


Figure 4.2: Distribution of ARP types according to purpose and solution types.

(solution dimension) is concerned with searching for suitable component designs to a design problem at hand, we call it "*Architecture configuration identification*" ARP. Thus, the combination of both dimensions, three types in the purpose dimension and four types in the solution dimension, leads to 12 types of ARPs in total. In Section 4.3.2, we present examples for the types of ARPs.

We further analyzed these types of ARPs into question variations, which are presented in Section 4.3.3 with a set of quoted examples. We used this classification for creating a sample to answer RQ4 (which types of ARPs in Stack Overflow do practitioners consider architecture relevant) in section 4.4.

Cross Architecture/Programming Posts (CAPPs) are posts with questions which could be either asked to perform a programming or an architecture activity. The answers to these questions provide information about technology solutions relevant for architecting but also for programming. Examples are posts which are about searching or selecting programming tools (e.g. unit testing frameworks, IDEs). These posts could be of interest to both the architect and the programmer: The architect is concerned with the impact of programming tools on the productivity and quality of the software or how tools integrate with frameworks and programming technologies (languages, libraries, etc.), while the programmer is interested in the programming features of tools. This type of posts are not analyzed further within the next section. Nevertheless, they are of interest to be addressed in a future work.

Summary on the types of posts Based on our analysis described in Section 4.2, we categorized 1,653 (65.8%) posts to be programming posts, 769 (30.6%) posts as ARPs, and 89 (3.5%) posts

as CAPPs. Fig. 4.2 shows the distribution of the different types of ARPs within our sample. The results show that the majority of the ARPs (91.4%) has one single purpose; either solution identification or evaluation. In addition, most of the ARPs (449 posts, 58.4%) use technology bundles as their solution type. When comparing the distribution of posts between the evaluation and identification posts, we can observe that the combined solutions post types within the solution identification posts are more than double of the combined solutions post types within the evaluation posts. On the other hand, technology bundle post types within the evaluation posts are nearly 30% more than the technology bundle post types within the solutions identification post.

4.3.1.1 Compact Types of Architecure-relevant Posts

In the previous section, we classified ARPs into 12 types according to their purpose and discussed architectural solutions. While this classification provides a detailed overview on the types of ARPs, a compact classification could provide a more comprehensive and summarized overview on the types of ARPs. Moreover, detailed analysis and machine learning algorithms require a bigger number of posts in order to produce useful and high quality results. In our main types of ARPs (the 12 types of ARPs), some types of ARPs have a limited number of posts. For example, we found only 4 instances of posts for the *Multi-purpose technology features ARPs*. A lower number of posts would negatively influence the quality of classification results.

Thus, in this sub-section, we aggregated several types of ARPs together to create a summarized overview on the types of ARPs. To do this, we performed the following two steps:

1. *Merge features and configuration ARPs*: We merged types of ARPs, which involve features and architecture configuration (i.e. features identification, features evaluation, architecture configuration identification, and architecture configuration evaluation) into a single type of ARP. We called this type "*features and configurations*". During our analysis, we found a relationship between features and architecture configurations ARPs. For example, in some feature identification posts, users explain a components design to use the technology features to solve a design issue.
2. *Re-assign multi-purpose and combined ARPs*: According to our initial analysis in Fig. 4.2, we observe that multi-purpose and combined solution ARPs represent minority (<20%) of ARPs. We re-assigned all multi-purpose and combined solution ARPs to one of the following three types of ARPs: *"technology identification", "technology evaluation" and "features and configuration"*. We determined the closest type by analyzing each post individually. We considered the majority of knowledge and discussions in the question and answers as the main factor to decide on the closest type of ARP.

In summary, the compact types of ARPs merge the 12 types of ARPs into three main types: *"technology identification", "technology evaluation"*, and *"features and configuration"*. The merging and re-assignment of the 858 ARPs resulted into 282 "*technology identification*" ARPs, 291 "*technology evaluation*" ARPs, and 285 "*features and configuration*" ARPs.

4.3.2 Examples for Architecure-relevant Posts

To clarify the different types of ARPs. We present in this section examples for six types of ARPs. To make the section readable and comprehensive, we do not present examples for the combined

Table 4.2: Example for technology identification ARP, stackoverflow.com/questions/4473567

Question: "We have cloud-hosted (RackSpace cloud) Ruby and Java apps that will interact as follows: 1. Ruby app sends a request to Java app. Request consists of map structure (...). 2. Java app analyzes data and sends reply to Ruby App. We are interested in evaluating both messaging formats (...) as well as message transmission channels/techniques (...). Our criteria: 1. Short round-trip time. 2. Low round-trip-time standard deviation. (...). 3. High availability. 4. Scalability"
Answer: "We have decided to go with BSON over RabbitMQ. We like BSON's support for heterogeneous collections and the lack of the need to specify the format of messages up-front. We don't mind that it has poor space usage characteristics and likely poorer serialization performance than other message formats since the messaging portion of our application is not anticipated to be the bottleneck."

Table 4.3: Example for technology feature identification ARP, stackoverflow.com/questions/1935040

Question: "I am looking into using WCF for a project which would require the ability for people to upload large files (64MB-1GB) to my server. How would I handle this with WCF, possibly with the ability to resume uploads. (...) I wanted to test out JSON via WCF. How would this affect the file upload? Can it be done from JSON, or would they need to switch to REST for the upload portion?"
Best answer: "If you want to upload large files, you'll definitely need to look into WCF Streaming Mode. (...) With Streaming, you can define either one-way streaming (for uploads only, for downloads only) or bidirectional streaming."
Answer-2: "MTOM is optimized to handle large binary data."
Answer-3: "You can use webHttpBinding with TransferMode streamed and a single Stream parameter or Stream response (as appropriate) for large file up/downloads"

Table 4.4: Example for architecture configuration identification ARP, stackoverflow.com/questions/2897513

Question: "I have been experimenting recently with Silverlight, RIA Services, and Entity Framework using .NET 4.0. I'm trying to figure out if that stack makes sense for use in any of my upcoming projects (...) but I'm struggling to decide how an application on top of this stack should be architected. (...) So, my questions: What is the best location for business logic (rules, validations, behaviors, authorization) in an application using this stack? Are there any guidelines published at an architectural level for using this stack? My questions pertain to large, complex, and long-lived applications."
Answer: "Our team is in the process of implementing a Silverlight app on top of the RIA stack. We've decided to build a domain model on top of the RIA entities. Additionally, we elected to follow the MVVM pattern to model UI interactions. So far, I've noticed the following benefits: 1. Domain classes are a nice place to put business logic including complex validations. 2. Domain classes use the RIA entities and context as interface to data store. 3. Domain classes are modeled after business concerns"

and multi-purpose ARPs. These types of ARPs (combined and multi-purpose) combine similar questions and answers from the other types of ARPs. Additional examples for all types of ARPs are available in Appendix A.

1. *Technology identification ARP*: The post in Table 4.2 stated a design problem by describing a subset of the architecture design of an existing system. The design consists of two applications communicating with each other in a cloud environment. Additional details on the architecture design (e.g., technologies, data structure) are given. The user then described the need to evaluate possible messaging technologies to decide on the communication between both applications, considering prioritized quality attribute requirements. Other users proposed answers to solve the design issue. At the end of the discussion, the user who posted the question posted an answer to describe the taken ADD "go with BSON over RabbitMQ", including the reason for taking this decision.

2. *Technology feature identification ARP*: The post in Table 4.3 ask about the ability to use the WCF technology, when uploading a big amount of data (up to 1 GB) to the server. The different answers proposed different technology features. For example, the best answer propose using the WCF Streaming Mode technology feature. Moreover, the answer explained the feature in details.

3. *Architecture configuration identification ARP*: The post in Table 4.4 asked about the best components design when using technologies "RIA Services, and Entity Framework". More specifically, it asked about the responsibility assignment ADD of the business logic. Another user proposed an answer describing a component design based on the MVVM (Model-view-view-model) architecture pattern.

4. *Technology evaluation ARP*: The post in Table 4.5 describes an architecture, which consists of three components. 1) Client applications, which submit requests. 2) Messaging broker queue, which handle messaging and notification. 3) Server applications, which listen and handle requests. The post also describes the sequence of communication. The question in post is concerned with determining the best technology for the messaging broker under the constrain that development of other components is done using C# language. Two technologies (ActiveMQ and MSMQ) are compared. The best answer recommended ActiveMQ and listed its benefits. Moreover, the second answer recommended MSMQ and listed its benefits. Finally, the third answer proposed using a third technology ZeroMQ.

5. *Technology feature evaluation ARP*: The post in Table 4.6 discusses choosing between different features for data representation. There are several options for data representation in .Net platforms (e.g. DTOs and STE). The question would like to know the suitable option (i.e. technology feature) for a certain situation, which include a client server structure with technologies (WPF, WCF and MS SQL). The answers mentioned benefits and drawbacks of using both DTOs and STEs. The first and best answer do not recommend STEs, while the second answer recommend STEs.

6. *Architecture configuration evaluation ARP*: The post in Table 4.7 evaluates a proposed layered architecture. In details, it discusses adding two service layers to offer data and functionality as services in two different layers. Both answers did not recommend the proposed architecture, because of its complexity for the described scenario.

Table 4.5: Example for technology evaluation ARP, stackoverflow.com/questions/32851

Question: "I'm working on a messaging notification system (...) The libraries will be written in C#. (...) My architecture will look something like app -> message broker queue -> server app that listens, dispatches all messages to where they need to go, and handles the life cycle of those long-lived messages -> message broker queue or topic -> listening apps. (...) Which message broker should I use?"
Best answer: "Some more benefits of ActiveMQ include -great support for cross language client access and multi protocol support. -excellent support for enterprise integration patterns (...) The main downside you mention is that the ActiveMQ broker is written in Java; but you can run it on IKVM"
Answer-2: "Pros for MSMQ. -It is built into Windows -It supports transactions, it also supports queues with no transactions. -It is really easy to setup. (...)"
Answer-3: "Take a look at zeromq. It's one of the fastest message queues around."

Table 4.6: Example for technology feature evaluation ARP, stackoverflow.com/questions/5407673

Question: "we should use DTOs or Self-Tracking Entities in our application? (...) We have a standard n-tier application with a WPF/MVVM client, WCF server, and MS SQL Database. Users can define their own interface (...) The Database layer spams multiple servers/databases (...) Our resources are limited (time, manpower, etc)"
Best answer: "If I understand your architecture, I think it is not good for STEs because: Models are used on both the client-side and server-side for validation (...) The main advantage (...) of STEs is their tracking ability but the tracking ability works only if STE is used on both sides"
Answer-2: "All STEs are POCOs, .Net dynamically add one layer to it for change tracking. Use T4 templates to generate the STEs. Pros for STE - You don't have to manually track the changes (...) Cons for STE - STEs are heavier than POCO, because of dynamic tracking"

Table 4.7: Example for architecture configuration evaluation ARP, stackoverflow.com/questions/2498796

Question: "We develop mostly low traffic but highly specialized web applications. Normally we use L2S, EF or nHibernate as access layer and then throws Asp.Net MVC to it and in which for normal crud operations we query the ISession/DataContext directly but for more advanced functions/side effects we put it in a some kind of service layer. Now, i was think about publishing the data through OData (WCF Data Service) and query that from the controllers (...) and publish the service operations through a WCF service (...). What advantages/disadvantages does this architecture poses?"
Best answer: "If you have direct line-of-sight to your database and it's your own application's database, then there is no reason to put WCF Data Services in the middle (...) Now, if you need to expose data over your http endpoint for other applications to consume (...) then definitely an OData endpoint may help and WCF Data Services is the simplest way to create one."
Answer-2: "Don't Do it. (...) this is a over-engineered approach. (...) Separation of concern is an illusion here - you replace a highly optimized domain model with a simplified data layer."

Table 4.8: Example for the Explicit Technology Solutions Searching ARP Variation, stackoverflow.com/questions/7837189

Question: "Does anybody know of a ESB written in Node.JS (...) I only need the following features for now: Content based routing, AAA, Logging, Monitoring"
Answer: "Take a look at SwarmESB, I haven't tried it yet, but it looks pretty interesting."

Table 4.9: Example for the Implicit Technology Solutions Searching ARP Variation, where the solution is a technology bundle, stackoverflow.com/questions/3400139

Question: "Is there a good way to handle multi-node and multi-core concurrency in Java where the main goal is to leverage the most CPU cycles as possible out of the cluster?"
Answer: "Depends on what you are doing and your budget you might want to look into (...) Norbert, GridGain, Terracotta."

4.3.3 Variations from Types of Architecture-relevant Posts

To further clarify the type of posts presented in the previous section, and to support the study replication, it is important to relate the post types with the different examples of each type. For this purpose, we present in this section the most common variations for the types of architecture-relevant posts (ARPs). The presented variations cover more than 85% of the ARP questions in our sample. We organized the variations of ARPs according to the purpose of questions (solution evaluation and solution identification). Under each variation, we specify relationships to solution types. Moreover, we include examples from posts from our sample for each variation. We provide additional examples in Appendix A.

4.3.3.1 Variants of Solution Identification ARPs:

1. *Explicit Technology Solutions Searching*: asks about solutions with a known type (e.g. framework). The solution type is mentioned explicitly in the question, with a set of selection criteria. This variant come only with technology bundles. Table 4.8 shows an example for a post, which ask for an Enterprise Service Bus (ESB) library with certain specification (e.g. it uses technology NodeJS), and supports technology features like routing and logging.

2. *Implicit Technology Solutions Searching*: searches for certain technology capabilities, which could be either technology features or bundles. The type of solutions are not mentioned explicitly within the post question. However, the solutions and their types are mentioned within the answers. Table 4.9 shows an example for a post, where a user search for a technology solution to perform multi-threading on multiple cores using Java. In the question, he did not specify the type of the solution (i.e. technology bundle or technology feature). However, the answer provides technology bundle solutions (e.g. Terracotta) as recommendations for solutions. Another example for this variation is provided in Table 4.10, where

Table 4.10: Example for the Implicit Technology Solutions Searching ARP Variation, where the solution is a technology feature, stackoverflow.com/questions/393580

Question: "Is there a way to setup authentication (ala "Basic Authentication") without actually setting up an SSL Certificate? I'd also like to do this in REST or regular SOAP WCF Services"
Answer: "Use TransportCredentialOnly security mode"

Table 4.11: Example for the Alternative Technology Solutions Searching ARP Variation, stackoverflow.com/questions/10156388

Question: "At the moment I have a solution that uses ZeroMQ to exchange protocol buffer payloads. The protocol buffer method of serialization is bound to stay as it is, but I can replace ZMQ with a more convenient option. (...) My requirements (...) -Support for Java, C++ and C# primarily, and Python, Ruby etc. -High performance. -Non Viral license, no GPL, AGPL etc. (...) What options can you think of other than ZMQ for this setup?"
Best answer: "Have you considered something like Storm or Spread?"
Answer-2: "You should probably have a look at Netty. It's a high performance Java NIO server framework"

Table 4.12: Example for the Solutions Recommendation for Requirements and Implementation of Conceptual Design ARP Variation, where the recommended solution is a technology bundle, stackoverflow.com/questions/2675793

Question: "Which network protocol to use for lightweight notification of remote apps? I have this situation (...) Client-initiated SOAP 1.1 communication between one server and let's say, tens of thousands of clients. Clients are external, coming in through our firewall, authenticated by certificate, https, etc.. They can be anywhere, and usually have their own firewalls, NAT routers, etc (...) They could be in a corporate/campus network, DSL/Cable, even Dialup"
Best Answer: "The two big parties on multi-tier development in Delphi are components4developers with their kbmMW product (...) and RemObjects with their product RemObjects SDK (...) In your complex environment, multi-cast UDP might not cut it, but from a overhead perspective it is unbeatable"

a user ask for a technology solution to have authentication without SSL and using WCF technology. The user did not specify explicitly the type of the solution (Technology bundle or feature). However, the answer recommend a certain technology feature (e.g. TransportCredentialOnly).

3. *Alternative Technology Solutions Searching*: searches for alternative or equivalent technology solutions to a known solution. Table 4.11 shows an example for a post, where a user searches for an alternative technology to ZeroMQ. He would like to replace ZeroMQ with another technology. However, he has requirements regarding the selection of a technology (e.g. support for programming language and good performance). The answers proposed several alternatives. Some of them justify their recommendations with facts about the features provided by technologies.

4. *Solutions Recommendation for Requirements and Conceptual Design*: asks for recommended solutions to a requirement scenario or implementation solutions for previously taken conceptual design. The requirements or conceptual design are mentioned in the question section, and the solutions might involve technologies, architectural configurations, as well as combined solutions. We present two examples for this variation in Tables 4.12 and 4.13. In Table 4.12 the user describes the requirements and explain the design of the system, which involves several clients and a server. The user asks about the most suitable protocols for communication. The answers discussed several solutions (e.g. kbmW and multi-cast UDP). Table 4.13 shows an example for this variation, where the user asks about possible solutions for an asynchronous data processing scenario. The answer propose architecture configurations as a solution, which consists of components (e.g. Windows service, database), and their sequence of communication.

Table 4.13: Example for the Solutions Recommendation for Requirements and Implementation of Conceptual Design ARP Variation, where the recommended solution is an architecture configuration, stackoverflow.com/questions/3198781

Question: "a stock analysis program with close to 50 years of stock data (...) filters that are applied on any given day to see if anything falls out for a trade. We want to run this filter for each day of data we have for each stock. (...) We are figuring about 40 hours or so to run the report on our entire data set (...) Does anyone have (...) experiences with a web architecture that will support ultra-long asynchronous processes?"
Best Answer: "I would recommend a standalone Windows Service, Console App or similar with very careful lifetime controls and logging, which would run constantly and check (poll) for 'jobs to process' in a database, then update the database with results and progress information."

Table 4.14: Example for the Technology Independent Architectural Configuration Searching ARP Variation, stackoverflow.com/questions/1139203

Question: "For message-oriented middleware that does not consistently support priority messages (such as AMQP) what is the best way to implement priority consumption when queues have only FIFO semantics?"
Best Answer: "Given only FIFO support for a given single queue, you will of course have to introduce either multiple queues, an intermediary, or have a more complex consumer. Multiple queues could be handled in a couple of ways"

5. *Technology Independent Architectural Configuration Searching*: asks for recommendations about conceptual design best practices, which are independent of technology constraints. Table 4.14 is an example for this variation, where users discuss suitable architecture configuration for queues. The answer propose several options (e.g. multiple queues, intermediary). The problem and the proposed solutions are independent of the technology and could be generally applied on various technologies.

6. *Technology Specific Architectural Configuration Searching*: asks for recommendations about architectural configuration in a scope of technology solutions. Table 4.4 is an example for this variation.

4.3.3.2 Variants of Solution Evaluation ARPs:

1. *Comparing solutions*: compares two different solutions. It involves asking about benefits and drawbacks, as well as reasons, contexts and factors for selecting a solution over another. The compared solutions could be technology bundles as previously presented in Table 4.5. The solutions could also be technology features (i.e. comparing technology features with each other) as presented in Table 4.6. Finally, users compare architectural configurations to evaluate them regarding quality attributes. Table 4.15 presents an example, which compares two architectural patterns (publish-subscribe and push-pull) in context of a technology ZeroMQ. The answer clarified the differences between both patterns, and suitable situations for each pattern.

2. *Context Independent Solution Assessment*: assess an architectural solution independent of any context. The assessment is done through listing benefits and drawbacks, or through evaluating the solution qualitatively and/or quantitatively regarding quality attributes. We

Table 4.15: Example for comparing solutions variation for architectural configurations, stackoverflow.com/questions/17814436

Question: "difference between pub-sub and push-pull pattern in zeroMq (...) What is the difference between this two pattern if we ignore sink in push-pull pattern ? Is there a difference in how a message gets transfer, if yes what is the difference?"
Best Answer: "The difference is that a PUB socket sends the same message to all subscribers, whereas PUSH does a round-robin amongst all its connected PULL sockets. In your example, if you send just a single message from the root, then all the subscribers will receive it (barring slow subscribers, etc.) but only 1 worker. The pub/sub pattern is used for wide message distribution according to topics. The push/pull pattern is really a pipelining mechanism. Your push/pull example seems to be attempting to do load-balancing, which is fine, but req/rep might be better suited to that due to other issues."

Table 4.16: Example for context independent solution assessment variation for technology bundle, stackoverflow.com/questions/512807

Question: "We are facing a choice to use IBM MQ over SFTP for file transfer. I've heard advantages of such approach, but I've never see anyone actually using it for a large files. So main question: how well IMB MQ can handle transfer of large files (up to 100 MB)? Is it stable? It's from mainframe to UNIX server, if it does matter."
Best Answer: "I've used MQ with files up to 8GB in size, without incident. You have to allocate enough space for MQ to manage them, but it works."
Answer-2: "MQ itself offers message-based communication between programs. You can't use it directly to transfer files (...) What you should be doing if you want to involve MQ in your file transfers is use one of the file-transfer products that sit upon MQ. (...) I've personally used WMQFTE to transfer a file of 1000GB from an AIX machine to a Windows one, so I don't think your "large" 100MB files will be a problem"

Table 4.17: Example for context independent solution assessment variation for technology feature, stackoverflow.com/questions/1189420

Question: "I have been building a client / server app with Silverlight, web services, and polling. Apparently I missed the whole Duplex Communication thing when I was first researching this subject. (...) When researching the scalability, it appears as if there's conflicting opinions on the subject. (...) In short, does anyone have facts / benchmarks?"
Best Answer: "There is an inbuilt 10 connection limit on non-server operating systems (XP/Vista/Windows 7) (...) On the server OS side (2003/2008), there is no connection limit. However, on IIS6 (2003) each long running connection will take a thread from the threadpool, so you will run into a connection limit pretty quickly. On IIS7 (2008), async threads get suspended in a way that does not use up a thread, so 1000s of connections should be possible."

Table 4.18: Example for solution scenarios and context variation for technology bundle, stackoverflow.com/questions/4499510

Question: "What are zeromq use cases? could you write some examples of zeromq?"
Best Answer: "Please see the ZeroMQ blog – they regularly post usage stories about different deployments, language bindings, etc."
Answer-2: "Let's say you want to have a bulletin board of some kind. You want to allow only some people to see it (...) This can be done using the publisher/subscriber model of ZeroMQ. Now, let's say you need to send some asynchronous messages. That is, when a message is sent from system A and needs to get to system B, it is guaranteed to be delivered later (...) You can imagine a use case being SMS messages (...) any JMS compliant solution like ZeroMQ will allow you to reliably broadcast or send a message"

Table 4.19: Example for solution scenarios and context variation for architecture configuration, stackoverflow.com/questions/969964

Question: "When to use SOA (...) "The only time we'll use services is when we need async actions otherwise we'll use go direct to the data store" (...) it seems fairly logical as services work well in a publish subscribe model, but I was wondering in what other scenarios you should be looking to use SOA?"
Best Answer: "We expose services to our customers because they shouldn't be able to connect to the datasource directly. We expose services to ourselves because it's easier to spread them over different technologies using WCF. We expose services because we have different user interfaces for the same datasource."
Answer-2: "Another case to use services is when you want to integrate a heterogeneous technology stack. In other words, if your DB is postgres, but you have code in Java, Perl"

presented previously an example for assessing an architecture configuration in Table 4.7. Independent assessment could be done for technology bundles and technology features as well. Tables 4.16 and 4.17 present examples for each respectively. In Table 4.16, users discuss about the ability of the technology bundle "IBM MQ" to transfer large files over SFTP. Users share their personal experiences regarding the ability of the technology (e.g. I've used MQ with files up to 8GB in size, without incident). In Table 4.17, a user ask a question about the scalability of the webservices and polling features in the technology "Silverlight". The user asked specifically about benchmarks. The best answer provided several benchmarks regarding the connection limits.

3. *Solution Scenarios and Context*: asks about suitable use-cases, and real experiences, where

Table 4.20: Example for the "context dependent solution assessment" variation among a technology bundle, stackoverflow.com/questions/9502548

Question: "I'm going to be working on a project that involves a number of elements: ASP.NET MVC website, C# console application, iPhone App (...) I now need to add an API to the site to allow third parties to select, insert and update records. (...) Should I be going along the line of using the Web API? or because my other applications need a web service, should I stick with a WCF Service?"
Best Answer: "If you intend to do RESTful development then you will definitely want to use the ASP.Net Web Api (...)"
Answer-2: "You can certainly do fairly well with a WCF RESTful service. Or you could use ASP.NET MVC."

Table 4.21: Example for the "context dependent solution assessment" variation among a technology feature, stackoverflow.com/questions/3543835

Question: "Our analytic server is written in c++. It basically queries underlying storage engine and returns a fairly big structured data via thrift. A typical requests will take about 0.05 to 0.6 seconds to finish depends on the request size (...) there are a few options in terms of which Thrift server we can use in the c++ code, specifically TNonblockingServer, TThreadedServer, and TThreadPoolServer (...).How shall I decide which one fits my needs the best?"
Best Answer: "Requests that take 50-600 milliseconds to complete are pretty long. The time it takes to create or destroy a thread is much less than that, so don't let that factor into your decision at this time. I would choose the one that is easiest to support and that is the least error-prone."

Table 4.22: Example for the "context dependent solution assessment" variation among a technology bundle, stackoverflow.com/questions/10160463

Question: "Is decoupling web and database tiers via MSMQ necessary or overkill? I'm putting together a simple asp.net web control, that (...) inserts a record into a MSQL database (...) the page containing this control will recieve many thousands of hits in a small space of time and I'm worried about the performance (...) A solution (...) is to make the web control place a message into an MSMQ queue and have a Windows Service on the server periodically read the queue and do a batch insert. Does that sound like a sensible architecture"
Best Answer: "Processing requests offline in this way is a common pattern for when you are dealing with a high volume of requests. (...) The main factor is: do your callers need to see the response to their requests synchronously? (...) if it is acceptable that callers can be sent a response saying that their request will be processed offline (...) then using a queue will definitely benefit your overall architecture. The first thing that will be benefited is front-end availability issues."

the architectural solution is suitable to be used. This variation comes with either technology bundles or architecture configurations. Table 4.18 shows an example for a post, where the user asks for use-cases for the ZeroMQ technology bundle. One answer referenced other blogs, which have list of use-cases. Another answer give examples for possible use-cases to use ZeroMQ (e.g. developing a bulletin board using the publish/subscribe feature in ZeroMQ). Users ask about possible use-cases for certain architecture configurations. For example, users discussed about possible use-cases and useful scenarios for a Service-oriented-architecture (SOA) in Table 4.19. The best answer provided three use-cases for using SOA to support re-usability of components, interoperability between heterogeneous technologies, and to offer services to customers, while protecting the security of data.

4. *Context dependent solution assessment*: assesses the suitability of a proposed solution to a certain context. The context could involve requirements or existing system design. This variation could come among the three types of solutions (i.e. Technology bundles, technology features and architecture configurations). An example for a post in this variation about a technology bundle is presented in Table 4.20. The question in this post describes an architecture of a system, which involves several components (e.g. clients, server, mobile applications), and asks about evaluating certain technologies (e.g. ASP.Net Web Api) for implementing a service to external customers. Answers provide several recommendations. Table 4.21 presents another example for this variation. However, users discuss several options for technology features (e.g. TThreadedServer, TThreadPoolServer) in the Thrift technology. Answers determine the differences between the features and also recommend one of them for this specified context. Finally, users assess the suitability of architectural configuration to certain contexts. Table 4.22 provides an example for this, where users evaluate certain components design using queues and asynchronous communication.

5. *Technology Solutions Interoperability Assessment*: assesses the interoperability of different technology solutions. This variation comes to assess the interoperability between technology bundles. Table 4.23 shows an example for a post, where a user asked about the complexity of using both technologies Zookeeper and RabbitMQ togethor. The best answer specifies possible requirements (e.g. performance requirements with throughput more than 1000 operations per second), which both technologies should be combined with each other.

6. *Solution Definition and Analysis*: defines the main features of a certain solution, as well as to clarify its behavior. This variation comes with technology bundles and architecture configurations. Table 4.24 shows an example for a technology bundle definition post. The user in this post asks about the BizTalk technology bundle and if it applies the enterprise service bus architecture pattern. Answers clarify the features of Biztalk technology and how it relates to the enterprise service bus. Table 4.25 shows an example for an architecture configuration definition and analysis post. In this post, users discuss about the enterprise service bus (ESB) architectural pattern. The answers clarify the role of ESB and its goal in achieving quality attributes.

4.4 Practitioner Evaluation

In the previous sections, we presented and explained using examples the different types of ARPs. In this section, we determine the types of ARPs in developer communities, which practitioners consider as architecture relevant (This answers RQ4). We first discuss how practitioners classified

Table 4.23: Example for the technology solutions interoperability assessment variation, stackoverflow.com/questions/2669573

Question: "I'm evaluating some components for a multi-data center distributed system. We're going to be using message queues (via either RabbitMQ or Qpid) so agents can make asynchronous requests to other agents without worrying about addressing, routing, load balancing or retransmission.(...) ZooKeeper seemed like it might be a good fit. But I'm wondering if trying to use both ZK and message queuing is overkill. (...) Has anyone out there used a combination like this?"
Best Answer: "The primary consideration should be the performance characteristics of your system. ZooKeeper, like you said, is more than capable of implementing a task distribution system using a distributed queue, but zk currently, is more optimized for reads than it is for writes (this only comes into play in the 1000's of ops per second range). If your throughput needs are less than this, then using just zk to implement your system would reduce number of runtime components and make it simpler."

Table 4.24: Example for the solution definition and analysis variation for technology bundle, stackoverflow.com/questions/3355082

Question: "I am looking into architectural patterns, Enterprise Services Bus (ESB) precisely. Upon reading this article Enterprise Integration, and with little to no experience I am wondering if BizTalk has is a ESB or is it just a EAI (Hub/Spokes or Bus)?"
Best Answer: "BizTalk is punted by Microsoft as having ESB capabilities (...) However, the term 'ESB' covers a very broad field (...) there are weak points in BizTalk's claim to be comprehensive as an ESB"
Answer-2: "BizTalk is a messaging and workflow orchestration platform, on which you can build ESB behaviours and capabilities. To make this easier, and standardise ESB implementation on BizTalk, Microsoft released the BizTalk ESB Toolkit - a set of guidelines, patterns and code."

Table 4.25: Example for the solution definition and analysis variation for architecture configuration, stackoverflow.com/questions/3055713

Question: "Can someone explain an Enterprise Service Bus to me in non-buzzspeak?"
Best Answer: "Although ESB is based on messaging, it is not "just" messaging and not just a buzzword. (...) The ESB attempts to address these issues with -Run-time resolution of destinations/services/resources. -Location transparency. -Any-to-any connectivity and maximum connection density. -Architected for redundancy, horizontal scalability, failover"

posts on Stack Overflow (based on the types of posts presented in section 4.3) and how practitioners agree with our own classification of posts into ARPs and programming posts (section 4.4.1). Finally, we show how practitioners agree with our types of ARPs (section 4.4.2).

Participants not only classified posts into ARP or programming posts, but also indicated their confidence. For each post, participants responded as follows:

1. *Yes - Architecture Post*: It is clear that the post is ARP.
2. *Maybe - Architecture Post*: The post is probably architecture-relevant.
3. *Do not know*: It is unclear, if this is a programming or architecture post.
4. *Maybe - Programming Post*: The post is probably about programming.
5. *Yes - Programming Post*: It is clear that the post is programming post.

We then calculated two main measures to evaluate our classification presented in the previous section:

1. *Agreement* between our own classification of posts into ARP and programming posts and the classification of posts into ARP and programming post by practitioners (i.e., the percentage of posts for which researchers and practitioners agreed versus the total number of posts classified). We calculated in section 4.4.1 - Table 4.26 the agreement for each participant for ARPs and programming posts, and for all participants for each post type (Table 4.26 - row "All"). In addition, we calculated the agreement across all posts (ARPs and programming posts) for each participant (Table 4.26 - column "Total"). Furthermore, we calculated the agreement for the different types of ARPs in section 4.4.2, Figs 4.3 and 4.4.
2. *Confidence level of agreement*, which has been calculated based on two levels of confidence ("Maybe" and "Yes") for each post. We assigned "Maybe" a value of 0 and "Yes" a value of 1. We then summed up the number of posts with "Yes" for agreed posts (programming post, ARP, Total) for each participant. By comparing this sum to the total number of posts, we obtained the confidence in percent (e.g., if out of 20 ARPs 15 were classified with a confidence of "Yes" and 5 with "Maybe" then the overall confidence for the agreement of that participant on classifying ARPs was 75%). Similar to agreement, we calculated confidence for each participant and each type of post, but also for each post type across all participants (Table 4.26 - row "All") and for each participant across all types (Table 4.26 - column "Total") and for the different types of ARPs (section 4.4.2).

4.4.1 Agreement and Confidence among Participants

Table 4.26 shows the classification results of the different participants. Column "Posts Given/Answered" shows the number of posts given to the participants ("Given") and the number of classified posts ("Answered"), i.e., the "Given" posts excluding the posts, which the participants marked as "Do not know". The pure programming, architecture-relevant posts, and Total Agreement and Confidence columns show the percentage of agreement and confidence across two groups dimensions; participant and post type.

Table 4.26: Agreement and confidence for each participant

Par. ID	Posts Given / Answered	Pure Programming Agreement & Confidence		Architecture-relevant Agreement & Confidence		Total Agreement & Confidence	
1	108/99	95.4%	95.5%	83.6%	92.7%	88.9%	93.9%
2	109/102	91.5%	85%	76.4%	87.3%	83.3%	86.3%
3	109/106	93.5%	85.1%	71.2%	87.3%	81%	86.3%
4	105/97	97.8%	91.3%	70.6%	59%	83.5%	74.2%
5	109/109	90.4%	86.5%	76.9%	75%	83.7%	80.8%
6	109/109	92.3%	86%	71.1%	69%	81.73%	77.9%
7	110/89	93.2%	50%	84.4%	24%	88.8%	37%
8	110/110	100%	96%	73.2%	89.3%	85.71%	92.4%
9	101/91	95.6%	82.2%	95.6%	78.3%	95.6%	80.2%
10	101/101	95.6%	78.3%	57.1%	51%	75.79%	64.2%
11	102/94	69.6%	84.8%	89.6%	85.4%	79.8%	85.1%
All	1173/1107	92.3%	83.6%	76.9%	71%	84.3%	77.1%

4.4.2 Agreement across ARP Types

In order to explain the measures presented in Table 4.26 and to answer RQ4, we calculated the total agreement and confidence levels across the different ARP types, i.e., question purpose (solution evaluation, solution identification, and multi-purpose) and solution type (technology bundle, technology feature, architecture configuration, and combined), without differentiating participants. Fig. 4.3 and Fig 4.4 illustrate the agreements and confidence levels.

We can observe that ARPs, which were classified by the authors to be solution evaluation or multi-purpose posts have high agreement and confidence, while the ARPs, which were classified by the authors to be solution identification posts have a moderate agreement and confidence. On the other hand, ARPs, which were classified by the authors to involve technology bundles have high agreement and confidence, while the ARPs, which were classified by the authors to involve technology features have a lower agreement and confidence. Moreover, architecture configuration posts have the highest agreement among solution types. However, they have moderate confidence level.

4.5 Discussion

4.5.1 Interpretation of Results

The results of our study indicate that it is possible to identify technology-related AK in Stack Overflow. In detail, we found that most of the ARPs belong to one of the two types according to the purpose of the question: solutions identification and evaluation. Both are common software architecture design activities [HKN+07]. A possible interpretation could be that Stack Overflow users perform architecture design activities (architecture identification and evaluation) and when they face problems they ask questions for each design activity separately. This finding is a positive indicator for the suitability of capturing and reusing of AK for technology decisions from within Stack Overflow posts.

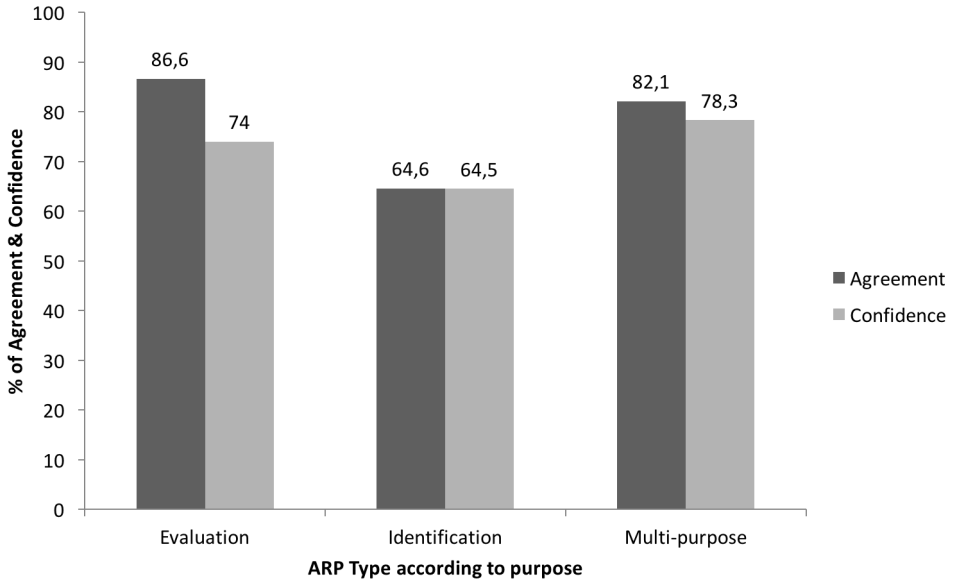


Figure 4.3: Agreement and confidence for ARP types according to the purpose of the question.

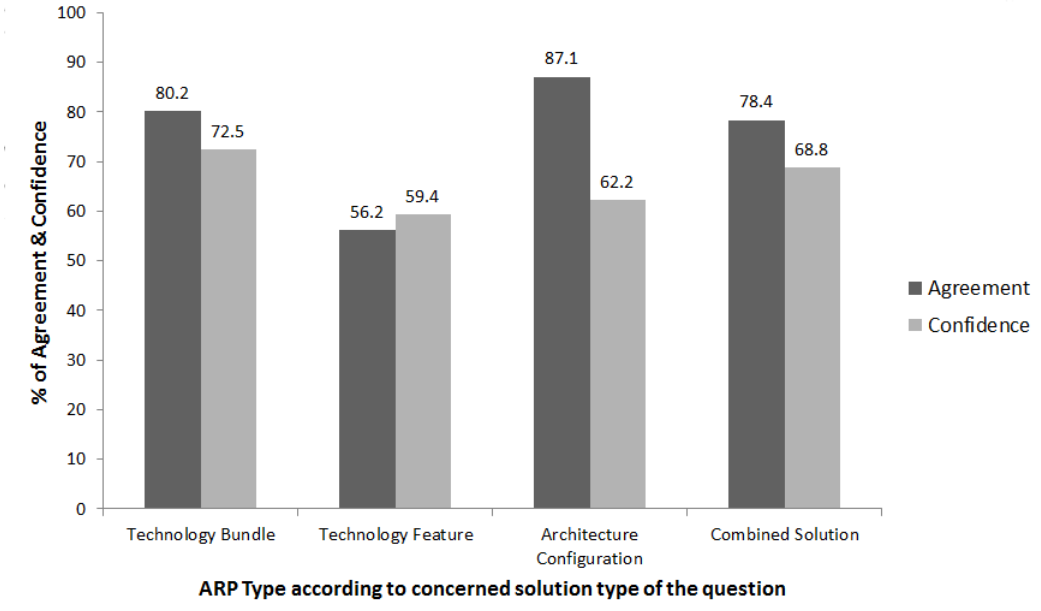


Figure 4.4: Agreement and confidence for ARP types according to the solution type.

The agreement of practitioners of classifying Stack Overflow posts as programming posts is higher than the agreement to classify posts as architecture post. This shows that the classification of posts performed by practitioners is more conservative than our own classification regarding their definition of software architecture. A reason could be the exploratory nature of our study and the keenness of the researchers to find all possible candidate ARPs. By analyzing the agreement across ARPs, we find that ARPs related to solution evaluation have a higher agreement and confidence level than solution identification posts. One explanation could be that due to the richness and importance of AK within the evaluation posts (e.g. benefits and drawbacks of solutions, decision rules). This type of knowledge embodies important factors for taking an ADD. On the other hand, the classification of some other types of posts differed among practitioners. For example, 56.2% of practitioners agree with our own classification of technology feature posts for being ARPs. These results show that there could be other important aspects within these types of posts which have influenced the classification decision of practitioners. The analysis and discovery of these aspects is subject to future work (see Chapter 9).

4.5.2 Implications of Results on further Research

Mining communities such as Stack Overflow for AK would provide several benefits over current AK management tools: Knowledge captured in Stack Overflow exists anyway (i.e., there is no need for an additional activity to document knowledge). The quality of the knowledge on Stack Overflow is ensured through evaluation of posts from users. Also, Stack Overflow provides a very large pool of knowledge and information that is constantly updated (unlike the typical expert systems that require manual population and maintenance).

Our result provide the first corpus of evaluated and classified architecture-relevant posts in one example of developer communities (i.e. Stack Overflow). This corpus supports further steps of research. In Chapter 5, we used this corpus to create an ontology of AK concepts in architecture-relevant posts. The corpus could be additionally used as a base for analyzing other developer communities. Our study has also an implication on the features, which architecture knowledge capturing and management tools should provide. A usage scenario could be a tool to browse and search for architectural knowledge within Stack Overflow with the ability to search for posts based on their design activity (identification or evaluation) and solution types (technology bundles, features or architectural configurations).

The types of ARPs provide two benefits, which support achieving our 3rd goal of the dissertation "Propose approaches to search for architectural information in developer communities":

Firstly: We used the compact types of ARPs to develop and test classification approaches, which automatically identify types of ARPs (see Chapter 7).

Secondly: We tried to determine types of ARPs, which support architecture design activities. For example, Kazman and Cervantes describe three generic incremental and iterative activities for finding solutions to architecture design problems [KC16]:

1. *Identify design concepts*: Types of architectural solutions (e.g. patterns, tactics, technologies) and candidate solutions (e.g. layers) in case of patterns, RabbitMQ for case of broker technologies) are identified. Technology identification posts might support this design activity, because they contain several recommendations for architectural solutions.

2. *Select design concepts*: Based on the benefits and drawbacks of alternative design concepts,

the most appropriate solutions are selected. Technology evaluation posts could support this activity, because they discuss the benefits and drawbacks of different architectural solutions.

3. *Instantiate architecture elements*: Selected design concepts are customized for the given design issue (e.g. for a layered pattern, we need to decide on the number of layers). Features and configuration posts could support this activity, because they discuss technology features and components.

In Chapter 8, we used the three compact types of ARPs to support searching for architectural information during each design activity.

5

Architecture Knowledge Ontology in Developer Communities

5.1 Research Question and Contributions 78
5.2 Research Process to Define Ontology 79
5.3 Architecture Knowledge Ontology in Developer Communities 82
5.4 Discussion 98

5.1 Research Question and Contributions

This chapter provides answer for **RQ5**:
RQ5: How can we represent and structure architecture knowledge from architecture-relevant posts in developer communities?

By answering RQ5, we support achieving the second goal of the dissertation "Explore developer communities for architecture knowledge" (see Chapter 1). To answer RQ5, we selected a sample of architecture-relevant posts (ARPs) from our Corpus of Stack Overflow ARPs (see Chapter 4). We considered the different types of ARPs (see Chapter 4) during our sampling. We performed then qualitative content analysis for textual segments in the sample of ARPs. In summary, we make the following **contribution**:

We present the first empirically-grounded ontology for AK in developer communities, using Stack Overflow (the most popular community) as an example. This ontology specifies how each AK concept in ARPs is composed. This bridges the gap between existing theoretical AK concepts (see Chapters 2 and 3) and their textual representation in developer communities. The ontology supports annotating and identifying AK concepts within ARPs. The developed AK ontology supports automating AK capturing. For instance, we used in Chapter 7 ontology-based classification approaches to identify and classify ARPs.

In this Chapter, we first explain our research process in Section 5.2 to define the ontology. In Section 5.3, we present and explain the ontology in details and using several examples. Finally, we discuss our analysis results in Section 5.4.

5.2 Research Process to Define Ontology

5.2.1 Data Gathering

When exploring Stack Overflow for architecture-relevant posts (see Chapter 4), we created a corpus of 858 ARPs, which were evaluated by practitioners for being architecture relevant. To answer RQ5, we randomly selected 105 ARPs from the available 858 ARPs. To better represent the population of ARPs, we selected ARPs using stratified random sampling [RR08], where ARPs have questions with different purposes and different types of solutions according to the 12 types of ARPs (see Chapter 4). This helps to cover a wide variation of ARPs, and consequently help us cover all AK concepts in ARPs.

5.2.2 Data Analysis

We applied qualitative content analysis [May14]. For the annotation process we used Atlas.ti, a qualitative data analysis software[1]. It helps human annotators annotate text segments within their context, and to assign them to categories. Also, the tool provides analysis and data modeling capabilities. Appendix A provides snapshots for using Atlas.ti during our annotation. We (the researchers[2]) annotated selected segments of text from our sample of ARPs. Each annotation is a tuple (s, c):

- s is a segment of annotated text. This could be a word or a clause or a sentence in a Stack Overflow ARP question or answer.
- c is an ontology class. An ontology class is an "explicit specification of a conceptualization" [Gru93].

The followed qualitative content analysis process consists of two phases as defined by Mayring [May14]: explication and structuring:

Explication: Explication aims at interpreting and comprehending text segments in ARPs to obtain a concrete definition for each ontology class. We started the annotation process with a list of ontology classes from known AK concepts (e.g. Quality attributes, ADD). The AK concepts were identified based on a review of existing AK models (e.g. [dBFL+07], [TGAS14]) (see Chapter 2). Moreover, we considered AK concepts about technology design decisions from our proposed AK model in Chapter 3. However, it was unknown how AK concepts are represented in ARPs. The first researcher (the author of the dissertation) annotated and analysed iteratively 40 randomly selected ARPs. Sentences in posts were assigned to one or more AK concepts. We excluded sentences which are not architecture relevant (e.g., sentences within an ARP about implementation details, source code). We then performed a semantic componential analysis (e.g., [Jac76]) for each annotated statement. During this process, selected clauses and words within the annotated sentences were further annotated and assigned to ontology classes. To correctly analyze the meaning of text in annotated sentences, external resources such as technology specifications and dictionaries have been used to understand some technology-specific terms.

[1] http://atlasti.com/

[2] The first researcher is the author of the dissertation. The second researcher is Matthias Galster from the university of Canterbury

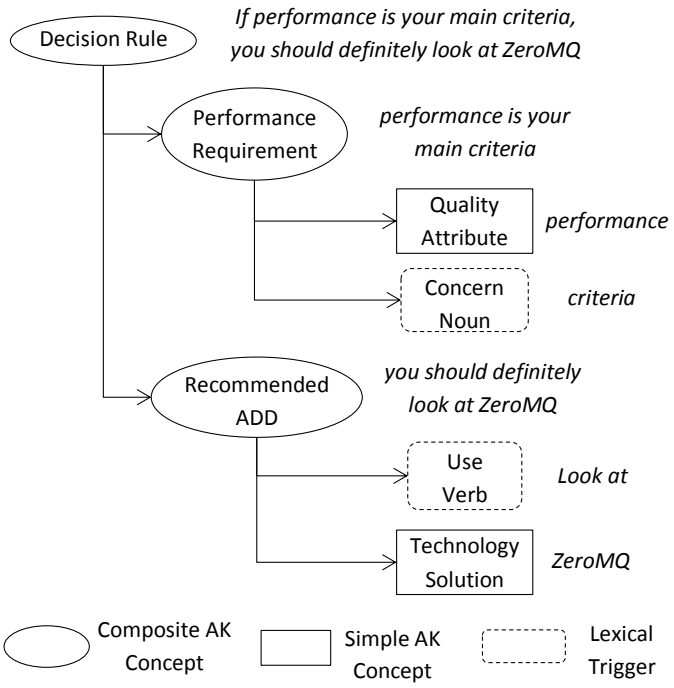


Figure 5.1: An example for an annotated statement in ARP

During annotation, the representation and structure of each ontology class started to emerge. Some of the AK concepts are represented in text as single words (e.g. quality attributes), we called these ontology classes *Simple AK Concepts*. On the other hand, we found AK concepts, which are represented in text as statements or clauses rather than as single words. We called them *Composite AK Concepts*. This type of classes is composed of other ontology classes in order to construct the semantics of statements or clauses. In addition to existing AK concepts, we discovered an additional type of classes, which are normal English words and not specifically related to a certain AK concept. Nevertheless, they deliver important meanings for composite AK concepts. The words in this class are further assigned to existing linguistic semantic categories as defined in WordNet [MBF+90], VerbNet [Sch05, Dix91]. We called these ontology classes "*Lexical Triggers*".

Fig. 5.1 shows an example of an annotated statement “If performance is your main criteria, you should definitely look at ZeroMQ” from an ARP[1] and annotated clauses and words in a hierarchical representation. Since the whole statement is an if-statement, it was annotated as a Decision Rule (*Composite AK Concept*), while the condition clause “performance is your main criteria” has been annotated as a Requirement (*Composite AK Concept*), and the result clause ”you should definitely look at ZeroMQ” as a Recommended ADD (*Composite AK Concept*). Within these two clauses, words with relevant semantics have been annotated. For example, “performance” is annotated as part of the if-condition clause for being a Quality Attribute (*Simple AK Concept*), and “Look at” is annotated as a Use Verb (*Lexical trigger*).

Based on this interpretation, the first researcher (the author of this dissertation) created an initial description of each ontology class, including definitions and examples for each ontology class and relationships between ontology classes. This description was the first version of our coding guide. Annotating statements within ARPs requires a subjective judgment of which a certain textual segment s refer to a certain ontology class c. Therefore, based on the first version of our coding guide, we conducted two inter-coder reliability tests. To conduct each test, we randomly selected

[1] stackoverflow.com/questions/17806977

a number of statements for each AK concept from the annotated posts by the first author. During each test, the second researcher[1] independently annotated the same statements and then both researchers met to reconcile disagreements and to discuss them to understand the main sources of disagreements. After each test, we revised the coding guide and updated our previously annotated statements. We also added a section "Differences between Concepts" to the coding guide to describe the differences between similar ontology classes. As a result of this phase, we obtained a mature coding guide with good agreement on the definition of each ontology class. This coding guide was used in the structuring phase (see next paragraph). Appendix A provides our coding guide.

Structuring: The structuring phase extracted the structure of the ontology across different types of ARPs based on the developed coding guide from above. We followed four steps:

1) Complete annotation: Based on the coding guide, we annotated the remaining 65 ARPs. In total, we created more than 3,800 annotations.

2) Count annotations for ontology classes: Using Atlas.ti, we counted the occurrences of annotations, which belong to each ontology class. We determined the occurrences of each ontology class and among each type of ARP. The frequencies of ontology classes for each type of ARPs are presented in Section 5.3.4.

3) Calculate co-occurrences: Using Atlas.ti, we captured annotated statements that appear in more than one ontology class, i.e., co-occurrences of annotations between ontology classes. This resulted in a hierarchy of ontology classes based on the co-occurrences of the annotations.

4) Determine significant ontology classes: We had two challenges: A) By the end of the annotation, we had more than 100 composite ontology classes. To refine our ontology and see the big picture, we merged several composite ontology classes, based on their frequencies and semantics into hierarchy of concepts to have 12 main composite ontology classes. B) We had more than 200 simple ontology classes and lexical triggers, which co-occur with the composite classes. To reach a concrete definition for each composite ontology class (identify children/composing ontology classes which are statistically significant). We calculated Pearson $\tilde{\chi}^2$ [Pea00] between composite classes and their co-occurring children/composing ontology classes. The test compares the observed frequencies of ontology classes in all annotations to the expected frequencies of each ontology class and its co-occurrences with other ontology classes. For example, for the two ontology classes "(REQ) Requirement" and "(QA) Quality Attribute", observed and expected frequencies for the following four situations are compared: i) Statements or clauses annotated as REQ intersect with annotations for QA. ii) Statements or clauses annotated as REQ intersect with annotations other than QA. iii) Statements or clauses annotated with class other than REQ intersect with annotations for class QA. iv) Statements or clauses annotated with class other than REQ intersect with annotations other than QA. We excluded co-occurrences with $\tilde{\chi}^2 < 10$ to ensure that all co-occurrences were statistically significant at $\rho < 0.05$. The significance test helps to understand the importance of each ontology classes. Section 5.3.3 presents the results of our significance test.

5) Final reliability test: We conducted the final inter-coder reliability test to assess the agreement on the definition of ontology classes. We focused on composite ontology classes, because they are represented in sentences and consist of other ontology classes, which make them challenging for an agreement. The test was done between the two researchers, and using the final version of the coding guide, and annotating another statements; 15% from the total annotated statements. We

[1]The second researcher is Matthias Galster from the university of Canterbury

Table 5.1: *Simple AK Concept* ontology classes

ID	Ontology Class Name	Examples of Words
TEC	Technology Solution	WCF, EJB, Netty, RabbitMQ
PAT	Architecture Pattern	REST, messaging, layer, SOA
QA	Quality Attribute	scalability, availability, throughput
COM	Architecture Component	server, backend, service, application
CON	Architecture Connector	read, send, write, communicate
COME	Component Element	interface, operation, record, job, call
COND	Connector Data	message, payload, information, data
PROB	Software Problem	SPOF, error, out of memory
FT	Feature Term	serialization, binding, deployment
ACTV	Programming Activity	debug, deploy, write code

calculated Cohen's Kappa reliability coefficient [Coh60] at 0.844, which indicates reliability and agreement beyond chance.

5.3 Architecture Knowledge Ontology in Developer Communities

By combining the hierarchical relationships between the referenced ontology classes across all annotations, we formed a Natural Language Ontology (NLO) [Ome01]. Our final ontology consists of 45 ontology classes (11 composite AK, 10 simple AK, 24 lexical triggers). We present the ontology in bottom-up (i.e. starting with simple ontology classes, which are presented with words, and end with composite ontology classes, which are presented using multiple sentences). We present first the simple AK and lexical triggers ontology classes in Section 5.3.1. We then discuss composite ontology classes using examples in Section 5.3.2. Finally, we discuss the relationships between ontology classes in Section 5.3.3.

5.3.1 Simple AK and Lexical Triggers Ontology Classes

Table 5.1 and 5.2 show lists of 'simple AK concept" and "lexical triggers" ontology classes. Each class is supported with examples of words from our analysis sample. In the following sub-sections, we explain each composite AK ontology class. We support our explanation with examples of annotated clauses or statements from our analysis sample. After explaining each ontology class, we discuss the relationships between ontology classes.

5.3.2 Composite Ontology Classes

5.3.2.1 (CONF) Architecture Configuration

The architecture configuration (CONF) ontology class represents part of an architectural model, which consists of one or more component names associated with an architecture connector verb or name. This ontology class supports building higher level composite ontology classes, which will be discussed in the following sections. For example, ontology class CONF supports describing an existing architecture or an existing software system when users explain a "Design Issue" (see

Table 5.2: *Lexical Trigger* ontology classes

ID	Ontology Class Name	Examples of Words
ADV	Advise Verbs	recommend, suggest, propose
AMT	Amount Term	many, more, thousands, lots
ASES	Assessment Verbs	evaluate, assess, appraise
BHV	Behavioral Verbs	run, perform, listen, process
CHR	Characteristic Nouns	advantage, weakness, pros, cons
CONC	Concern Nouns	requirement, criteria, demand
CONS	Constraint Nouns	constraint, limitation, restriction
DFF	Difference Noun	difference, distinction
DIF	Difficulty Adjectives	lightweight, complex, overkill
DIS	Discover Verbs	search, look, seek
FIT	Fit Verbs	fit, integrate, complement
FORC	Force Verbs	force, have to
LER	Learn Verbs	learn, acquire, teach
PROB	Problem Nouns	problem, obstacle, trouble
QUE	Question Word	which, what, when, how
REL	Rely Verbs	depend, implement, count on
SOL	Solution Adjectives	solution, alternative, option, choice
SPED	Speed Adjectives	fast, slow, heavy, quick
STAY	Stay Verbs	stay, stick, adhere, bind
SUPP	Support Verbs	offer, provide, supply, support
USE	Use Verbs	select, choose, use, prefer, go with
VAL	Value Adjectives	good, outperform, important
VS	Versus Preposition	versus, vs., against, contrast
WISH	Wish Verbs	need, require, want, demand, ask

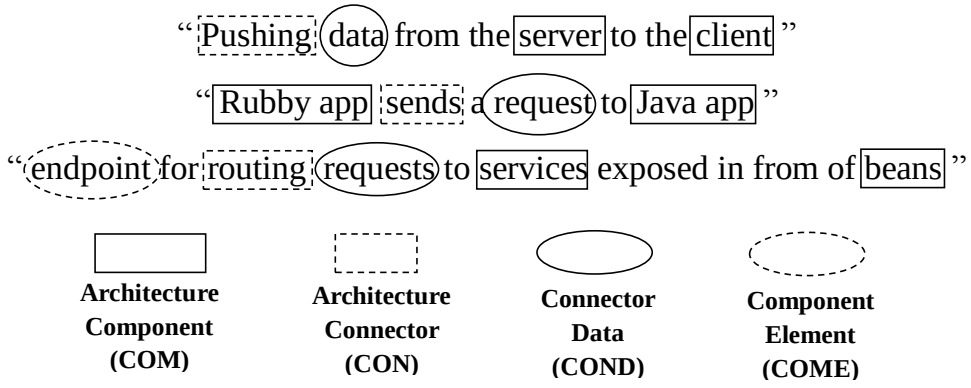


Figure 5.2: Examples of annotated sentences of architecture configuration (CONF) ontology class. Each sentence is further annotated with its composing ontology classes. The three sentences belong to three posts: stackoverflow.com/questions/12783677, stackoverflow.com/questions/4473567, and stackoverflow.com/questions/19758215

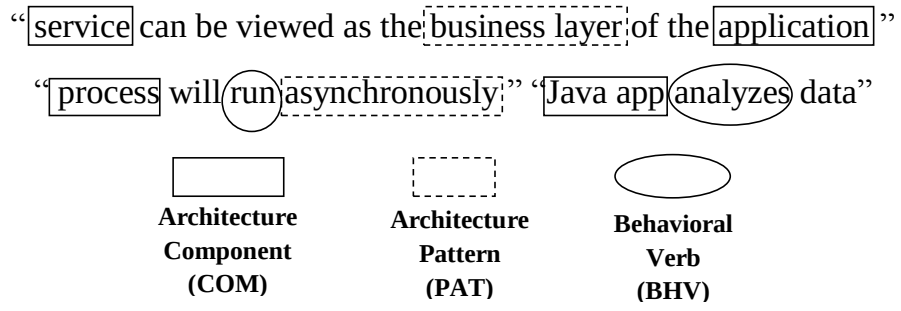


Figure 5.3: Examples of annotated sentences of component behavior (CB) ontology class. Each sentence is further annotated with its composing ontology classes. The three sentences belong to three posts: stackoverflow.com/questions/1582952, stackoverflow.com/questions/380052, and stackoverflow.com/questions/4473567

Section 5.3.2.4). Ontology class CONF is also used when recommending certain design decisions for a design issue to describe the recommended components design (see Section 5.3.2.9). Sentences from the architecture configuration (CONF) ontology class are composed of words, which belong to four simple AK ontology classes: COM, CON, COME, and COND. COM and CON are mandatory to create a CONF, while COME and COND are optional.

Fig 5.2 shows examples for annotated sentences, which belong to the architecture configuration (CONF) ontology class. The sentences are further annotated with their composing ontology classes COM, CON, COME, and COND.

5.3.2.2 (CB) Component Behavior

The component behavior (CB) ontology class describes the behavior of an architecture component. It gives an overview about the type of implemented logic and complexity. Sometimes internal operations are mentioned during the description. Similar to the CONF ontology class, this ontology class supports building higher level composite ontology classes, which will be discussed in the following sections. For example, ontology class CB supports describing the behavior of an architecture when users explain a "Design Issue" (see Section 5.3.2.4). Sentences from the component behavior (CB) ontology class are composed of words, which belong to the simple AK ontology classes: COM, BHV, and PAT. Ontology class COM is mandatory to compose CB. One of the two ontology classes BHV or PAT must appear to express the behavior of the component.

Fig 5.3 shows examples for annotated sentences, which belong to the component behavior (CB) ontology class. The sentences are further annotated with their composing ontology classes COM, BHV, and PAT.

5.3.2.3 (EX) Existing System

The existing system (EX) ontology class describes part of an architecture of an existing software system. It additionally describes the possible problems in the system. This ontology class supports describing the architecture of a software system when users explain a "Design Issue" (see Section 5.3.2.4). To express an existing system (EX), an architectural configuration (CONF) composite ontology class (see Section 5.3.2.1) is required, which consists of components (COM) and con-

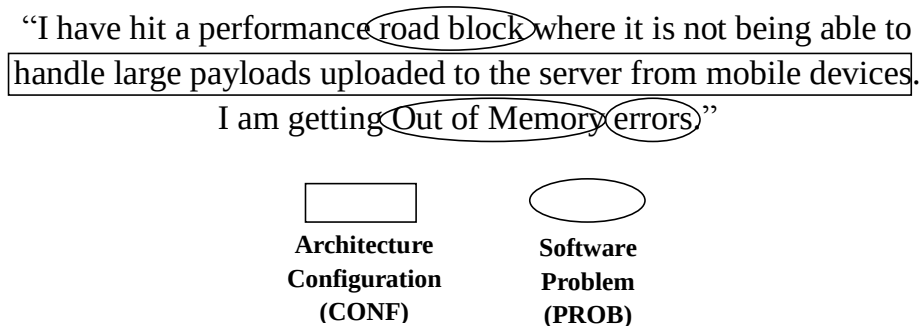


Figure 5.4: Example for an annotated sentences of existing system (EX) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belong to the post: stackoverflow.com/questions/13016406

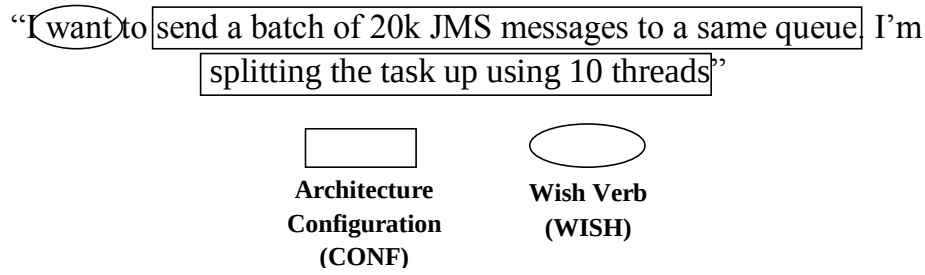


Figure 5.5: Example for an annotated sentences of design issue (DI) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belong to the post: stackoverflow.com/questions/4741713

nectors (CON) simple AK ontology classes. Moreover, EX uses the lexical trigger ontology class PROB to express a problem in an existing system (e.g. a performance or a scalability problem).

Fig. 5.4 shows an example for annotated sentence, which belong to the existing system (EX) ontology class. The sentence is further annotated with its composing ontology classes CONF and PROB.

5.3.2.4 (DI) Design Issue

Design issue (DI) describes design problems, which users submit in the question section in an ARP. Design problems are described through the architecture configurations of a planned design, or the architecture configuration design of an existing software system. This ontology class is one of the high-level ontology classes in the question section of an ARP. Sentences from the design issue (DI) ontology class are composed of sentences, which belong to the CONF, CB and EX composite ontology classes as explained in Sections 5.3.2.1, 5.3.2.2, and 5.3.2.3. Moreover, DI commonly uses two lexical trigger: USE, and WISH. At least one of the two lexical triggers USE or WISH must appear in a DI to express the notion of a need.

Fig 5.5 shows an example for annotated sentence, which belong to the design issue (DI) ontology class. The sentence is further annotated with its composing ontology classes CONF and WISH.

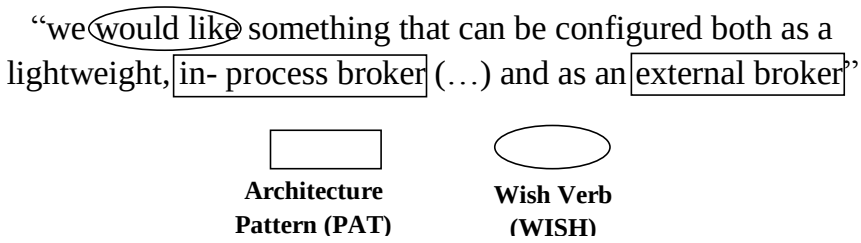


Figure 5.6: Example for an annotated sentences of the technology feature requirement, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/14342430

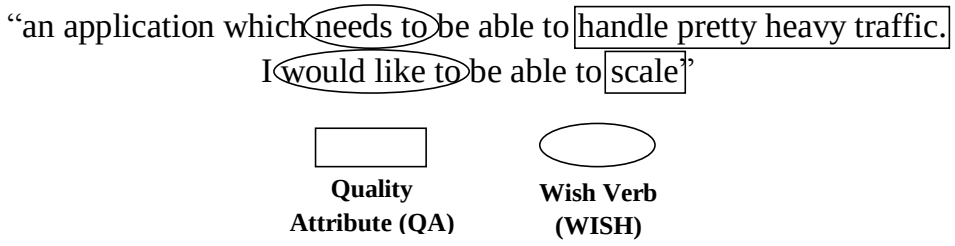


Figure 5.7: Example for an annotated sentences of the quality attribute requirement, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/2567254

5.3.2.5 (REQ) Requirements and Constraints

The REQ ontology class is one of the high-level ontology classes, which appear in both question and answers sections of an ARP. This ontology class has several sub-types:

1. *Requirements*: They are basic requirements of a system. We found two types of requirements:

   (a) *Technology feature requirement*: They are specific functionality, which technology solutions should fulfill. Sentences from this type are composed of the FT and PAT simple AK ontology classes. Moreover, they uses two lexical trigger: CONC, and WISH. At least one of the two lexical triggers CONC or WISH must appear. Fig 5.6

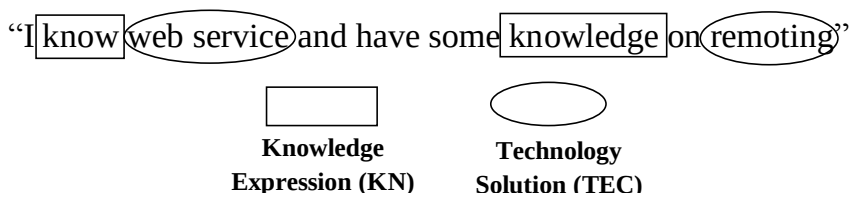


Figure 5.8: Example for an annotated sentences of the team skills constraint, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/1426249

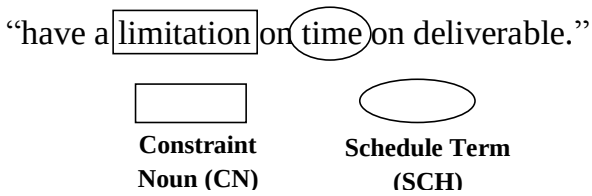


Figure 5.9: Example for an annotated sentences of the development time constraint, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/807962

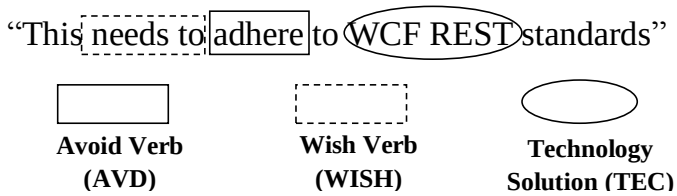


Figure 5.10: Example for an annotated sentences of the solution constraint, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/12783677

shows an example for annotated sentence, which belong to this sub-type. The sentence is further annotated with its composing ontology classes WISH and PAT.

(b) *Quality attribute requirement*: They are requirements regarding the quality attributes (e.g. performance, security) of the system. Sentences from this type uses two lexical trigger: CONC, and WISH. Quality attributes is expressed directly using the simple AK ontology class QA. Fig 5.7 shows an example for annotated sentence, which belong to the quality attribute requirement. The sentence is further annotated with its composing ontology classes WISH and QA.

2. *Constraints*: They are restrictions, which limit the selection of a solution. We found three types of constraints:

   - *Technical skills constraint*: This is a limitation regarding the technical skills of the team, who will implement the solution. Sentences from this type use the lexical trigger KN to express the knowledge about certain technology solutions TEC. Fig. 5.8 shows an example for annotated sentence, which belong to this sub-type. The sentence is further annotated with its composing ontology classes KN and TEC.
   - *Development time constraint*: This is a limitation regarding the schedule and available time and budget for implementing a solution. Sentences from this type use the lexical triggers CN and SCH to express the limitation regarding time and effort. Fig. 5.9 shows an example for annotated sentence, which belong to this sub-type. The sentence is further annotated with its composing ontology classes CN and SCH.
   - *Solution constraint*: This is a limitation regarding the characteristics of the architectural solution. Sentences from this type use the lexical triggers AVD and TECH to express the limitation regarding a certain technology. Fig. 5.10 shows an example for annotated sentence, which belong to this sub-type. The sentence is further annotated with its composing ontology classes AVD and TECH.

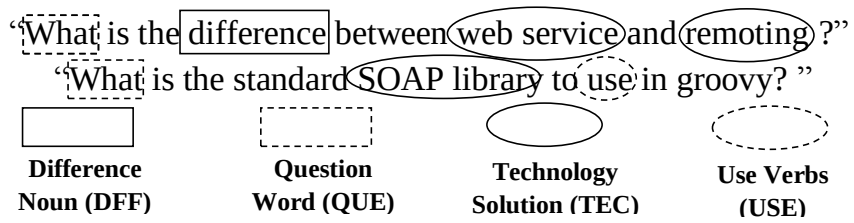


Figure 5.11: Example for an annotated sentences of the user request (UR) ontology class. The sentence is further annotated with its composing ontology classes. The sentences belong to the posts: stackoverflow.com/questions/1426249 and stackoverflow.com/questions/10621648

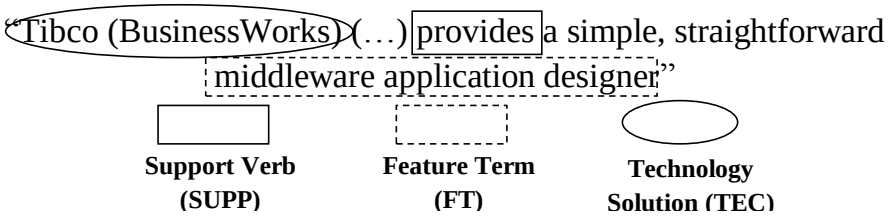


Figure 5.12: Example for an annotated sentences of the development feature, a sub-type from the technology feature (FEAT) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/1429318

The representation of a REQ ontology class is different, when it appear in the question and the answers sections of an ARP. Requirements in the question section are gathered, expressed in details and listed in points. On the other hand, requirements in the answers section are separated for each other, and individually addressed.

5.3.2.6 (UR) User Request

(UR) User Request exists in ARP question in a form of questions or needs. It complements design issue, requirements and constraints by showing the purpose of the question (evaluation or identification). User requests could be further classified according to the variations of Evaluation ARPs (see Chapter 4). This ontology class is one of the high-level ontology classes in the question section of an ARP. Sentences from the user request (UR) ontology class are composed of several lexical triggers QUE, VS, USE and DFF lexical triggers. Moreover, UR commonly uses the simple AK ontology class TEC. The usage of lexical triggers in this ontology class UR specify the type of the post (evaluation or identification). For example, VS and DFF are commonly used to evaluate a technology solution. While the USE lexical trigger come usually in solution identification posts.

Fig 5.11 shows examples for annotated sentences, which belong to the user request (UR) ontology class. The sentences are further annotated with its composing ontology classes TEC, DFF, USE and QUE.

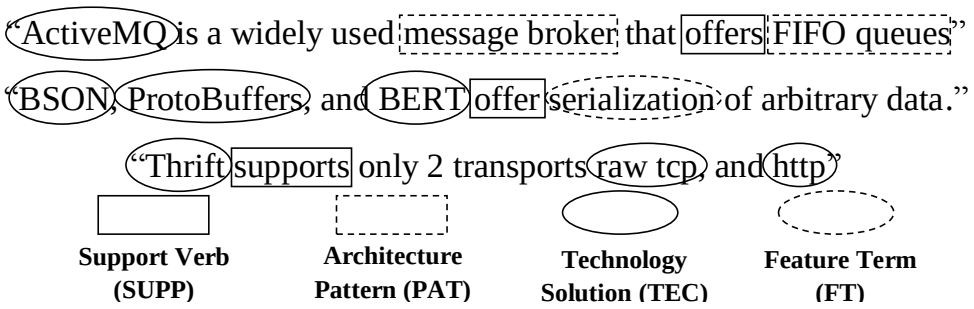


Figure 5.13: Example for an annotated sentences of the behavioral features, a sub-type from the technology feature (FEAT) ontology class. The sentences are further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/10375137, stackoverflow.com/questions/4473567, and stackoverflow.com/questions/7583132

5.3.2.7 (FEAT) Technology Feature

The FEAT ontology class is one of the high-level ontology classes, which mostly appear in answers sections of an ARP. This ontology class has several sub-types, which align with the types of technology features (see Chapter 3). All sub-types of FEAT ontology class use the lexical triggers SUPP and REL associated with one or more technology solutions TEC to express relationships between technology solutions and technology features. Other simple AK ontology classes (PAT, FT) appear based on the type of technology feature (Development or Behavioral).

1. *Development feature*: They present technology features, which support the development of a software (e.g. documentation, development tools) as explained in Chapter 3. They are expressed through certain programming activities (e.g. debugging) or programming features and tools (e.g. code generation). Fig. 5.12 shows an example for annotated sentence, which belong to this sub-type. The sentence is further annotated with its composing ontology classes TEC, SUPP and FT.

2. *Behavioral features*: They are features about the run-time behavior of the technology. They are expressed through technology specific component and class names, as well as their implemented architectural patterns or their relationship with other technologies. Fig. 5.13 shows an example for annotated sentence, which belong to this sub-type. The sentence is further annotated with its composing ontology classes TEC, SUPP, PAT and FT.

5.3.2.8 (ASTA) Technology Benefits and Drawbacks

The ASTA ontology class is one of the high-level ontology classes, which mostly appear in answers sections of an ARP. As explained in Chapter 3, ASTA concepts are main factors, which support architects decide between different technology solutions (i.e. taking technology design decisions). Sentences which belong to the ASTA ontology class are distinguished through the extensive usage of adjectives and adverbs in combination with technology features and quality attributes. The adjectives or adverbs are used to express the advantages or disadvantages of certain technology solutions or features. ASTA ontology class could be further classified according to their influence on certain technology features (i.e. development features versus behavioral features):

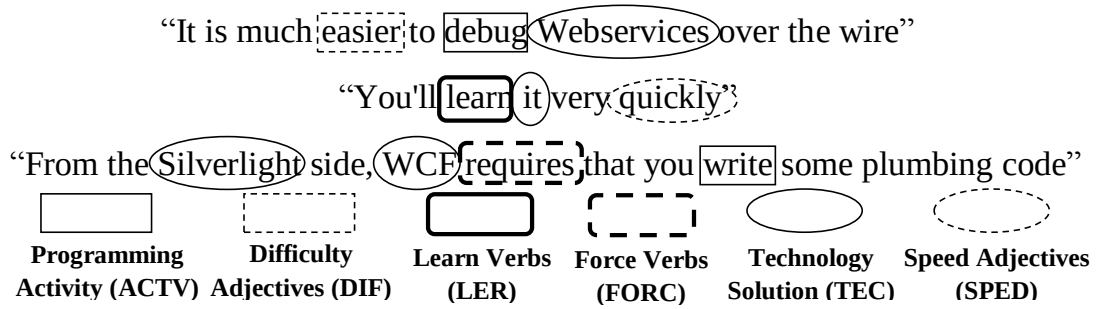


Figure 5.14: Example for an annotated sentences of the development ASTA, a sub-type from the technology benefits and drawbacks (ASTA) ontology class. The sentence is further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/100993, stackoverflow.com/questions/807692, and stackoverflow.com/questions/1582952

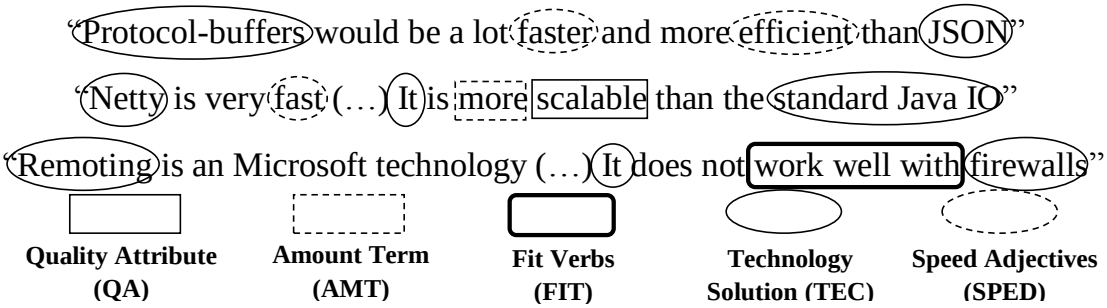


Figure 5.15: Example for an annotated sentences of the behavioral ASTA, a sub-type from the technology benefits and drawbacks (ASTA) ontology class. The sentences are further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/4473567, stackoverflow.com/questions/5145129, and stackoverflow.com/questions/1426249

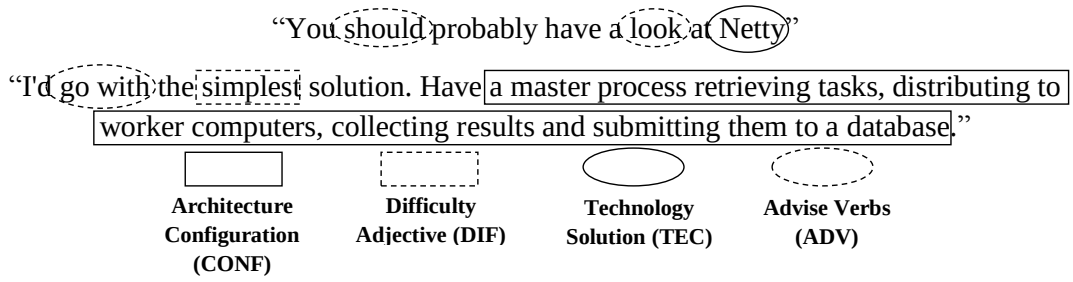


Figure 5.16: Examples for annotated sentences of the ADD. The sentences are further annotated with their composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/10156388 and stackoverflow.com/questions/9535421

- *Development ASTA*: It is used to express benefit or drawback regarding certain development features (e.g. how good is the documentation or community of a technology). Users use the ACTV ontology class to express the complexity of conducting certain development activity. Moreover, users use the LER lexical trigger to show the complexity of learning a certain technology solution. To differentiate between benefits and drawbacks, five lexical triggers are used: FORC, DIF, SUPP, CHR, and AMT. Fig. 5.14 shows examples of development benefits and drawbacks. The sentences are further annotated with their composing ontology classes.

- *Behavioral ASTA*: It is used to express benefit or drawback regarding certain behavioral features. Behavioral ASTA could be further categorized according to the quality attributes (e.g. availability, performance). One common way to express behavioral ASTAs is the usage of the simple AK ontology class QA and VAL lexical trigger. Alternatively, certain verbs could be used instead of using QA. For example, SPED lexical trigger is used to reference performance ASTA, and FIT lexical trigger is used to reference interoperability ASTA. Fig. 5.15 shows examples of development benefits and drawbacks. The sentences are further annotated with their composing ontology classes.

5.3.2.9 (ADD) Recommended design decision

The ADD ontology class is one of the high-level ontology classes, which mostly appear in answers sections of an ARP. ADDs are recommendations from users based on their experience or opinion for certain architectural solutions.

Recommendations for architectural solutions could be for technology solutions, or architectural configurations or technology features. An ADD contains either TEC, or CONF, or FT to refer to an architectural solutions. Lexical triggers ADV, VAL and USE are commonly used to provide the meaning for recommendation. Fig. 5.16 shows examples for ADD on different architectural solutions. The sentences are further annotated with its composing ontology classes.

5.3.2.10 (DR) Decision Rule

The DR ontology class is one of the high-level ontology classes, which mostly appear in answers sections of an ARP. DR are conditional recommendation for architectural solutions. The rule condition involves other ontology classes such as requirements and constraints (REQ), architectural

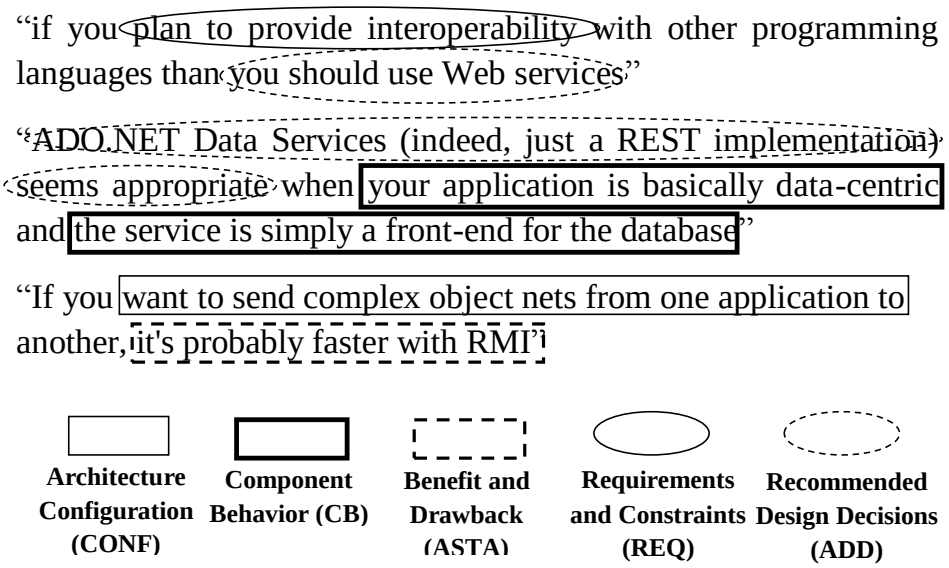


Figure 5.17: Example for an annotated sentences of DR. The sentences are further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/1426249, stackoverflow.com/questions/1582952, and stackoverflow.com/questions/100993

configuration (CONF), and component behavior (CB). Recommendations involve recommended (ADD)s for certain technology solution or architecture configuration. Fig. 5.17 shows examples for (DR)s with different conditions and architectural solutions. The sentences are further annotated with their composing ontology classes.

5.3.2.11 (CASE) Technology Use-Cases

The CASE ontology class is one of the high-level ontology classes, which mostly appear in answers sections of an ARP. Technology Use-Cases (CASE) are either success or failure stories for the usage of technology solutions at certain contexts. The stories could be coming from personal experiences of users, or well-known examples for existing systems. The context associated with stories could include domain description, architecture configurations, infrastructure, and constraints. CASE is considered the most involved ontology class, which might include several other composite ontology classes like CONF, REQ, ASTA, and ADD. Fig. 5.18 shows examples for CASE on different architectural solutions. The sentences are further annotated with its composing ontology classes.

5.3.3 Relationships between Ontology Classes

In the previous sections, we presented each ontology class individually. In this section, we focus on the relationships between ontology classes.

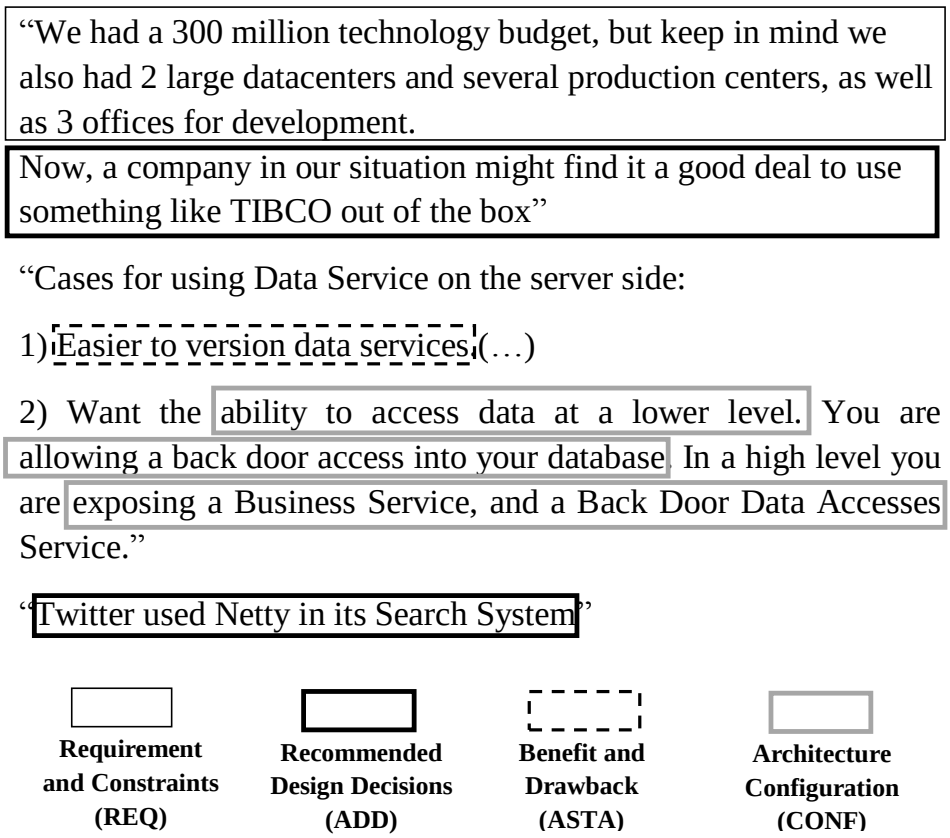


Figure 5.18: Examples for annotated sentences of CASE. The sentences are further annotated with their composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/1429318, stackoverflow.com/questions/2508361, and stackoverflow.com/questions/5145129

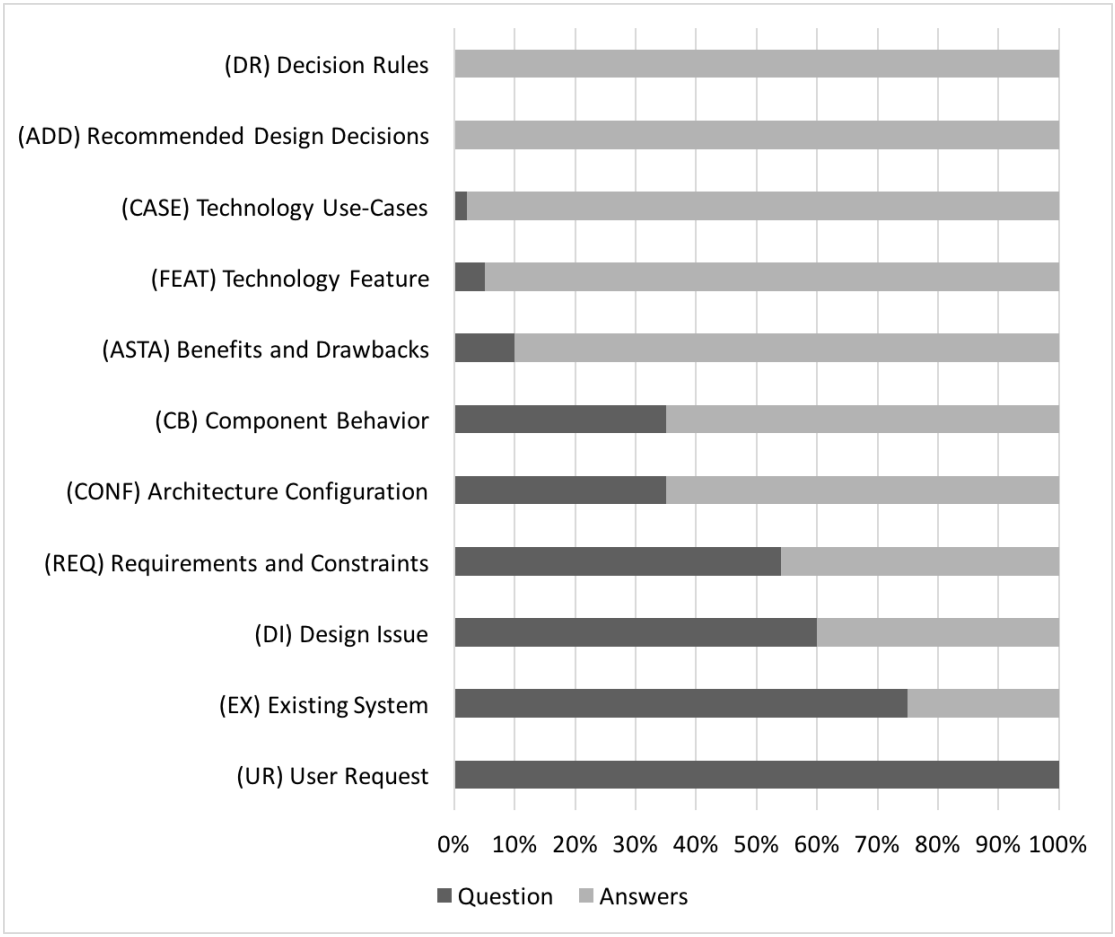


Figure 5.19: The percentages of occurrences for composite ontology classes within the question and answers sections of a Stack Overflow post

5.3.3.1 Ontology Classes in Post Structure

As explained in Chapter 2, a Stack Overflow post consists of a question section, and an answers section. To better understand the ontology classes, we identified to which section, would each composite ontology class mostly appear. We counted the number of annotations in our sample of ARPs, which appear in each section (i.e. either question or answers) of a Stack Overflow post. Fig. 5.19 shows a stack bar chart, which presents percentages about, where each composite ontology class usually appears. From Fig. 5.19, we can observe that UR, and EX appear mostly in the question section of a Stack Overflow post. On the other hand, FEAT, ASTA, ADD, DR, and CASE appear mostly in the answers sections of a Stack Overflow post. The ontology classes CONF, and CB appear in both sections, because they are building blocks for many other ontology classes (EX, DI, ADD, and CASE). Finally, ontology classes DI and REQ appear in both sections. Annotations for DI and REQ in the question section are bigger in their size, because users try to explain in details requirements or design issues. On the other hand, REQ and DI in the answers section reference certain problems or requirements as part of other ontology classes (e.g. REQ could appear as part of DR).

To explain how different ontology classes could construct a post, Fig. 5.20 shows an example for an annotated ARP. The annotations use the IDs of the composite ontology classes in Section 5.3.2. The question in the ARP stated a design issue (ontology class *DI*). The design consists of two applications communicating in a cloud environment (ontology class *CONF*). The user then described the need to evaluate possible messaging technologies to decide on the communication between both applications (ontology class *UR*), considering prioritized quality attribute requirements (ontology class *REQ*). One user (Answer 2) shared a success story (ontology class *CASE*) for a well-known system, which solves a similar problem. At the end of the discussion, the user who posted the question posted an answer (Answer 1) to describe the taken decision (ontology class *ADD*) "go with BSON over RabbitMQ", including the rationale for taking this decision, which include technology features (ontology class *FEAT*), their benefits and drawbacks (ontology class *ASTA*), and the logic behind choosing the technology among other alternatives (ontology class *DR*).

5.3.3.2 Significant Relationships between Ontology Classes

When discussing composite ontology classes in Section 5.3.2, we clarified some relationships between a composite ontology class, and its composing ontology classes. However, some ontology classes appear more often with others. As explained in Section 5.2, we determined the significant composing ontology classes, which appear with each composite ontology class using a $\tilde{\chi}^2$ significance test. Table 5.3 presents our results. In column "Composing Classes", we provide the significance $\tilde{\chi}^2$ value for each composing class from which the composite ontology class is constructed. The results of the significance test shows that some simple AK and lexical triggers ontology classes do not appear to be significant with any of the composite ontology classes, because of their lower number of occurrences.

By determining the significant relationships between ontology classes, we could propose more effective information extraction techniques (e.g. [WD10]) for architectural information in developer communities.

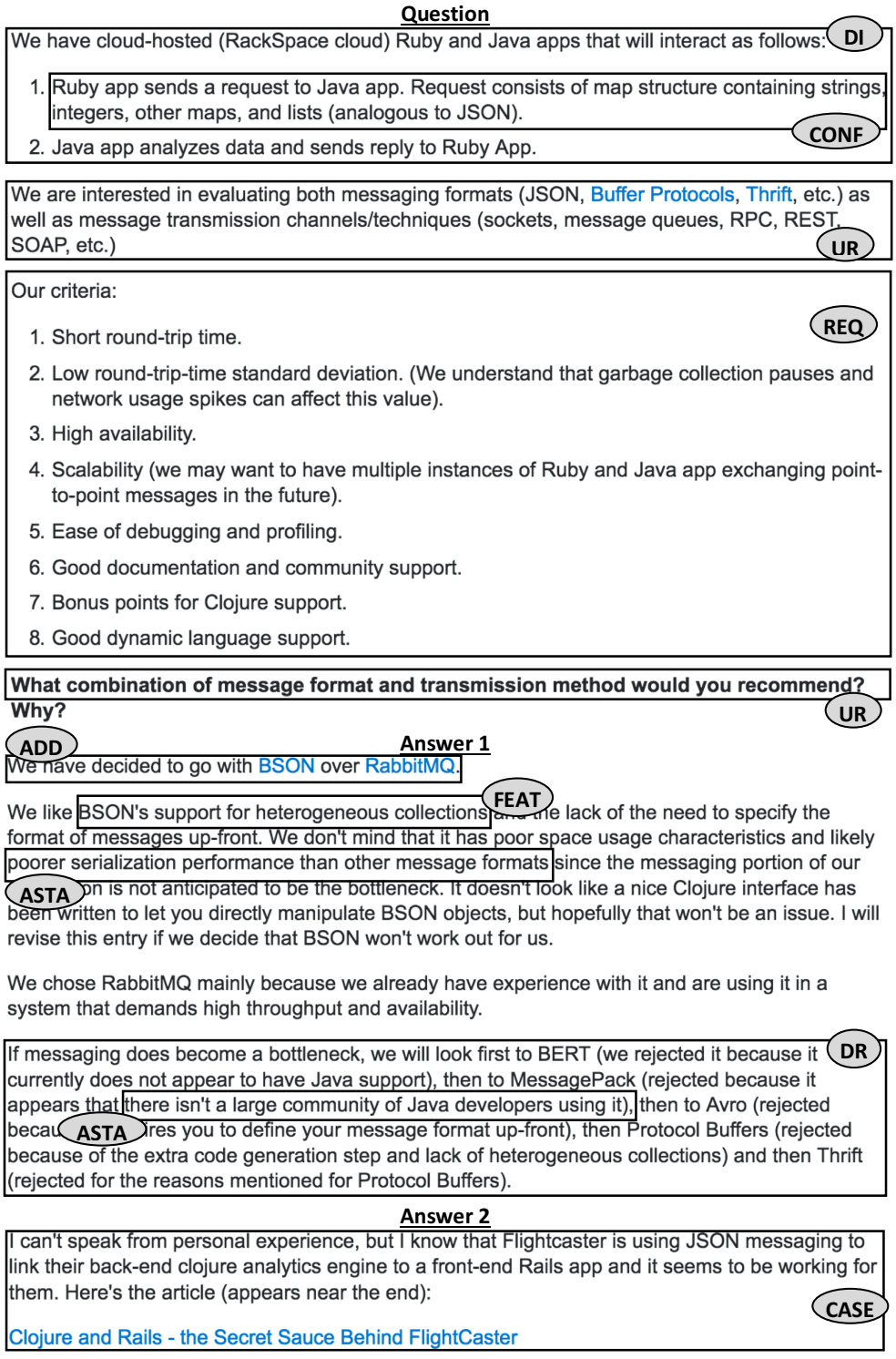


Figure 5.20: An example for an annotated ARP https://stackoverflow.com/questions/4473567

Table 5.3: Statistically significant composing ontology classes for composite ontology classes

(ID) Composite Ontology Class	**Composing Classes** ($\tilde{\chi}^2$)
(CONF) Architecture Configuration	CON(441.4), COM(326), COND(193.9), COME(41.7)
(CB) Component Behavior	COM(85.7), COME(64.6)
(EX) Existing System	PROB(158), COM(123.7), CONF(42), CON(35.8)
(DI) Design Issue	CONF(110), EX(95.3), CB(43), WISH(37.4), USE(14.7)
(REQ) Requirement and Constraint	WISH(213.7), QA(108.7), CONC(74.16), DIF(12.17)
(UR) User Request	QUE(324.16), TEC(238.8), USE(116), VS(42), DFF(37)
(FEAT) Technology Features	TEC(90), PAT(74.8), FT(44.1), REL(35.7), SUPP(13.6)
(ASTA) Technology Benefits and Drawbacks	VAL(130), DIF(102.7), QA(57.4), SPED(49.7), TEC(18.9)
(CASE) Technology Use-Cases	CONF(106), CONC(48.7), USE(45.8), ASTA(29.7), REQ(15.5), ADD(13.3)
(ADD) Recommended ADDs	ADV(355), TEC(72), USE(41), CONF(18.49)
(DR) Decision Rules	ADD(369.4), REQ(226.7), CB(85), ASTA(60.7), CONF(54.9), FT(12.65)

5.3.4 Distribution of Ontology Classes in Types of ARPs

In the previous sub-sections, we introduced the different explored AK ontology classes, and their relationships. In this section, we explore the occurrences and distributions of AK ontology classes in the types of ARPs, which we classified in Chapter 4 according to the purpose of the post and the used architectural solutions. The benefit of knowing the occurrences of ontology classes in the types of ARPs is two folds:

1. We will have a better definition and understanding for the types of ARPs. This will complement our qualitative analysis and definition (based on the purpose and type of solution) for each ARP as presented in Chapter 4.

2. Determining concrete relationships between the types of ARPs and the ontology classes support improving the search for architectural information. For instance, if an architect searches for architectural information to evaluate architectural solutions, the ASTA ontology class would be useful to him. It would be easier then to search in the types of ARPs, which contain the most instances of ASTAs. In Chapter 8, we propose and evaluate an improved searching approach. The approach considers the occurrences and distributions of ontology classes in the different types of ARPs.

We counted the number of annotations for each composite AK ontology class, and among the three compact types of ARPs (Technology Identification, Technology evaluation, and features and configurations). We decided to use the compact types of ARPs to ensure the availability of

Table 5.4: Number of annotations for each ontology class among the different types of ARPs

	Technology Evaluation ARPs		**Technology Identification ARPs**		**Features and Configurations ARPs**	
	Frequency	*Relative Frequency*	*Frequency*	*Relative Frequency*	*Frequency*	*Relative Frequency*
(FEAT) Technology Feature	92	2.49	47	2.04	23	0.96
(CASE) Technology Use-Cases	25	0.67	8	0.35	14	0.58
(CB) Component Behavior	18	0.49	10	0.43	19	0.79
(CONF) Architecture Configuration	33	0.89	23	1	24	1
(ASTA) Technology Benefits and Drawbacks	227	6.13	89	3.87	32	1.33
(REQ) Requirements and Constraints	31	0.83	26	1.13	9	0.37
(DR) Decision Rule	58	1.56	12	0.52	8	0.33
(DI) Design Issue	40	1.08	25	1.08	29	1.21
(EX) Existing System	9	0.24	6	0.26	0	0
(ADD) Recommended Design Decision	61	1.65	54	2.34	46	1.92
(UR) User Request	83	2.24	35	1.52	27	1.12

sufficient annotations for each type of ARP. Table 5.4 shows the number of annotations for each ontology class and across each type of ARP. We also calculated the number of occurrences relative to the total number of posts for each type of ARP. Fig. 5.21 shows a matrix bubble chart, which compares the relative occurrences for each composite AK ontology class among each of the three compact types of ARPs. The relative number of annotations supports comparing the ontology classes in each type of ARP. Moreover, the calculated relative occurrences of ontology classes support estimating quotas for the types of ARPs, when searching for architectural information. We used these values in Chapter 8 in an approach to improve the search for architectural information in developer communities.

5.4 Discussion

5.4.1 Interpretation of Results

The proposed ontology of natural language concepts related to AK on Stack Overflow makes the concept of *architecture knowledge* concrete. The concepts in our ontology and the ontology classes contain elements that are also mentioned in other literature related to AK (see Chapter 2). For example, De Boer et al. [dBFL+07] have mentioned decisions or design alternatives as AK concepts, but our study presents the first empirically-grounded ontology from a concrete developer community (i.e. Stack Overflow). Previous AK models try to cover the most AK concepts of the architecture domain. Elements in these models include decisions or rationale which are high level. The proposed ontology makes the concepts in AK models functional through a concrete

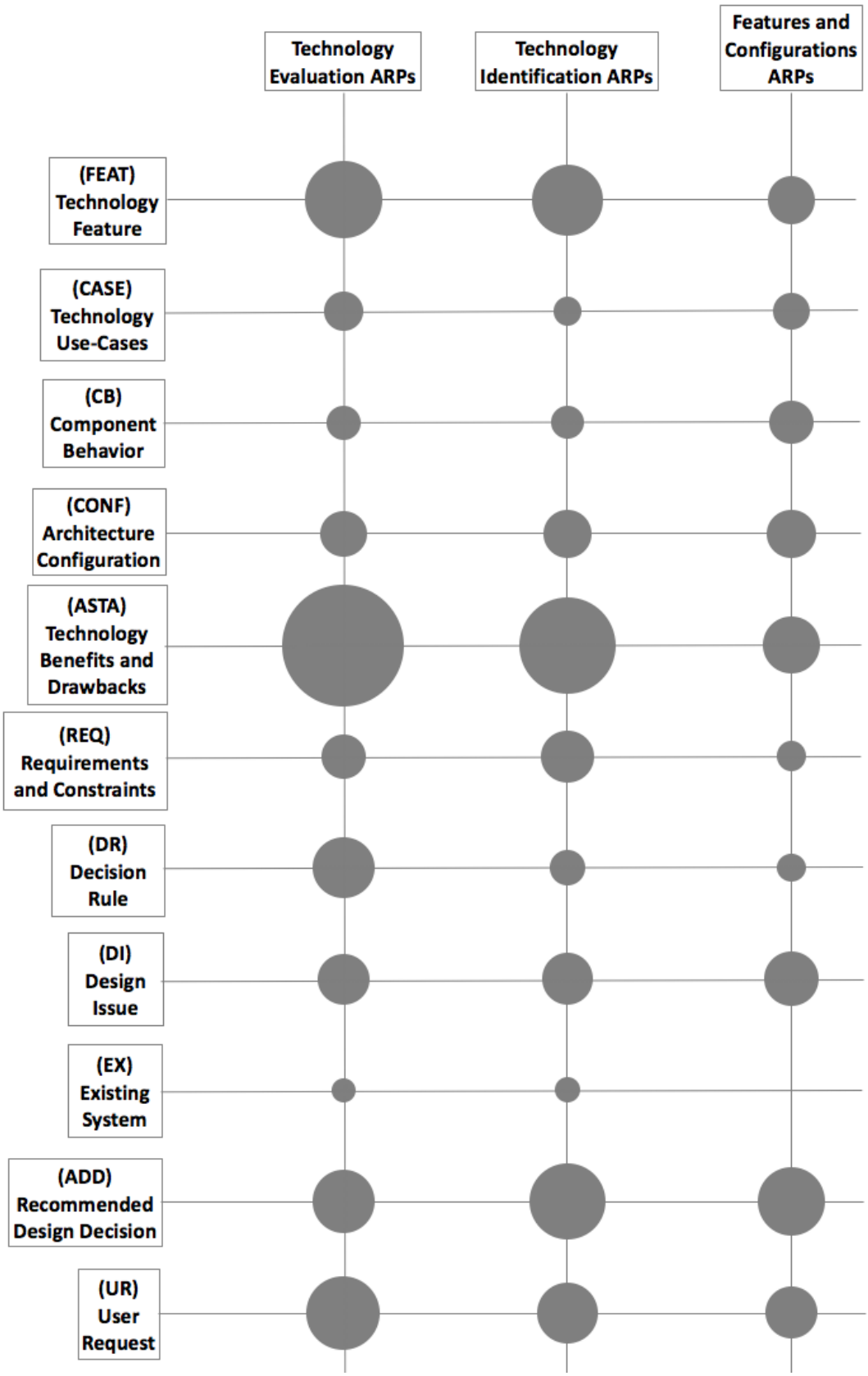


Figure 5.21: The relative occurrences of ontology classes in each type of ARP

specification for each concept. This supports building tools to automatically identify architecture-relevant information from a body of text in developer communities.

5.4.2 Implications of Results on further Research

The ontology supports ontology-based approaches for AK capturing. First, ontology-based approaches could specify AK concepts using a formal language (e.g., using OWL). Moreover, ontology-based approaches support methods for information extraction (e.g. [WD10]), which could automatically identify textual segments that belong to a certain AK concept. This is quite useful to automate the population of architectural knowledge in knowledge repositories. Moreover, information extraction is a main building block in ontology-based information retrieval (e.g. [FCL+11]). This could support developing ontology-based search engines, instead of only relying on keywords-based search. For instance, searching for keywords does not consider relationships between search terms. Without considering the relationships between semantic concepts in text, higher level concepts such as AK concepts cannot be easily identified. In Chapter 7, we used an ontology-based classification approach to identify and classify ARPs in developer communities. Based on the classification of posts, we complemented a keywords-based search with the semantics of posts, which improved the effectiveness of search as presented in Chapter 8.

Part III

Solutions for Architecture Knowledge Acquisition

6

Scenarios and Perspectives for Using Developer Communities during Architecture Design

6.1 **Research Questions and Contributions** 102
6.2 **Research Process** 103
6.3 **Perspective of Practitioners on AK Sharing and Reuse from Developer Communities** 105
6.4 **Scenarios for Searching AK in Developer Communities** 109
6.5 **Discussion** 112

6.1 Research Questions and Contributions

This chapter provides answers for **RQ6** and **RQ7**:
RQ6: What is the perspective of practitioners on AK sharing and reuse from online developer communities?

RQ7: How could architects acquire AK in developer communities?

By answering RQ6 and RQ7, we support achieving the third goal of the dissertation "Propose approaches to search for architectural information in developer communities" (see Chapter 1). To answer RQ6 and RQ7, we analyzed the literature and conducted interviews with ten practitioners from different companies. More details on the research process is explained in Sec. 6.2. In summary, we make the following **contributions**:

- We identify empirically and practically-grounded *benefits* and *problems* of, as well as *solutions* for, utilizing developer communities to share and acquire AK (see Sec. 6.3).
- We identify main *conceptual elements* (tasks and activities) of scenarios to search for AK during architecture design. We then identify concrete *tasks and activities* of scenarios to use AK from developer communities (see Sec. 6.4).

Our contributions in this chapter provide an empirically-grounded basis for AK tools to increase the usefulness of tools for practitioners. Moreover, our interviews with practitioners provided us

with real scenarios to search for architectural information in developer communities. Furthermore, the identified conceptual elements for a searching scenario support designing searching tasks for architectural information. Both the scenarios and model have been used to propose an approach to search for architectural information in developer communities (see Chapter 8).

In section 6.2, we describe our research process. In section 6.3, we present benefits, problems and solutions for using developer communities to find architectural knowledge. This answers RQ6. In section 6.4, we present our proposed conceptual model for a architectural information searching task, as well as our findings of use-cases and scenarios. This answers RQ7. Finally, section 6.5 discusses our findings.

6.2 Research Process

To answer our research questions, we conducted semistructured interviews with ten practitioners (see Sec. 6.2.1). We obtained two types of information:

- Perspectives of practitioners for using developer communities as a source of AK. We categorized practitioners' perspectives into benefits, problems and solutions (see Sec. 6.3). This allowed us to answer **RQ6**.
- Examples of scenarios for using developer communities as a source of AK and for sharing and reusing during architecture design. To identify the components of a scenario (tasks and activities), we analyzed these examples of scenarios using concepts from the field of software architecture and information studies (see Sec. 6.4.1). These tasks and activities were then checked by interviewees for applicability and completeness (more details are in Sec. 6.2.2). This answers **RQ7** (see Sec. 6.4.2).

6.2.1 Interview Study

We chose interviewees based on a) their practical experience as architects, b) the size of the companies they were working at (we considered practitioners from small, medium and large companies), and c) the diversity of experiences (we selected practitioners from organizations in different domains). All interviewees had a Bachelor's or Master's degree. Interviewees are listed in Table 6.1. Interviewees were located in different cities and countries, so interviews were done via phone, VoIP and face-to-face.

Interviews were semi-structured. We prepared an interview guide, which is available in Appendix B. Additional questions came up during the interviews based on the interviewee responses. Questions aimed at understanding a) the process of searching and using developer communities as part of the architecting process and architecture design activities, and b) concerns/problems of architects during searching for AK on developer communities and possible solutions. We recorded and transcribed the interviews. The average length of the interviews was 60 minutes. After transcribing interviews, inductive open coding content analysis [May14] allowed us to structure interview data along several categories: benefits, problems, solutions, and example scenarios to use AK from communities.

Table 6.1: Interview participants

ID	IT exp. (years)	Arch. exp. (years)	Domain	# Company employees	Role
I1	11	5	Manufacturing	>100,000	Solution architect
I2	11	4	Telecom	<1,000	Enterprise architect
I3	15	6	Games	>100,000	Solution architect
I4	12	4	Transportation	>100,000	Technology consultant
I5	11	5	Telecom	<500	Technology consultant
I6	14	3	Telecom	<1,000	Solution architect
I7	11	5	Finance	<1,000	Solution architect
I8	16	8	Telecom	>100,000	Solution architect
I9	12	3	Language processing	>100,000	Technology consultant
I10	7	2	Mobile apps	<500	Solution architect

6.2.2 Analysis of Scenarios

The examples of scenarios mentioned by practitioners involve two types of concepts: concepts related to architecture design and concepts related to information retrieval. Both types of concepts have been intensively studied in two different research fields: architecture design processes and information studies. Thus, in order to properly analyze the scenarios mentioned by architects, we first analyzed relevant literature in both fields.

In the field of information studies, several conceptual models [BH05] have been proposed to model the notion of a *Task* during information access. These models are used for analyzing and designing information retrieval systems (e.g. search engines). On the other hand, software architecture design processes (e.g. Hofmeister et al. [HKN+07], ADD-3.0 [KC16]) describe common steps or activities, which are performed by architects to design the architecture of a software system. Based on our literature analysis, we identified and related relevant concepts of both fields. The identified concepts structure scenarios to search for AK into three concepts: information-intensive design activities, information seeking tasks (ISTs), and information searching activities (ISAs) (see Sec. 6.4.1).

To answer RQ7, we identify information-intensive design activities, ISTs, and ISAs from the example scenarios obtained from the interviews. The combination of them creates an enormous number of possibilities for scenarios. To get a list of information-intensive design activities and ISTs, we analyzed architecture design processes. To make the analysis practically feasible, we focused on one software architecture design process and considered design activities defined in the Attribute-Driven Design (ADD-3.0) process [KC16] (see Chapter 2 for additional explanation). We chose ADD-3.0, because it aggregates the main design activities from the architecture

literature. Moreover, ADD-3.0 considers technology decisions and their influence during the design steps. This makes ADD-3.0 quite suitable to solve our research problem, which focus on technology design decisions, and their associated architectural knowledge. We analyzed ADD-3.0 steps to identify information-intensive design activities and to determine ISTs. This was done by reviewing the documentation of ADD-3.0 and analyzing the information needs of each step. To get a list of ISAs, we performed qualitative content analysis of the real scenarios mentioned by the interviewees. These concrete scenarios combine information intensive design activities, ISTs, and ISAs.

In order to evaluate the captured design activities, ISTs, and ISAs, we combined them into concrete scenarios, and in a follow-up asked interviewees through a questionnaire about the applicability and completeness of these scenarios. The questionnaire is available in Appendix B. Based on their answers, we refined our list of ISTs and ISAs. Sec. 6.4.2 presents information-intensive design activities, ISTs, and ISAs.

6.3 Perspective of Practitioners on AK Sharing and Reuse from Developer Communities

In this section, we present prespectives of practitioners to search for architectural information in developer communities. This answers RQ6. During the interviews, we found that practitioners rely mainly on their own *personal experience* when solving a familiar design problem. This confirm findings of other previous empirical studies on design decisions [HA11]. For unfamiliar problems (e.g. involving unfamiliar technologies or domains), our interviews found that architect consult other sources of AK. The other sources of AK mentioned by practitioners in our interviews were: *corporate knowledge repositories*, *help from an expert* (e.g. by getting help from another person in the organization or an external consultant), and open *developer communities* (e.g. Stack Overflow, Google groups). This means that architects indeed access developer communities for solving architectural problems. When investigating the perspective of practitioners on AK sharing and reuse from developer communities, we categorized findings based on the following categories:

- Benefits of using developer communities as a source of AK (Sec. 6.3.1).
- Problems architects face when searching for AK in communities (Sec. 6.3.2).
- Solutions (proposed by interviewees) to improve the search for AK (Sec. 6.3.3).

In the following subsections, we discuss these categories in more detail. Table 6.2 summarizes the distribution of the identified benefits, problems and solutions across interviewees. When explaining each of these categories, we will compare developer communities with other sources of AK.

6.3.1 Benefits of Using Developer Communities as a Source of AK

We identified the following benefits of developer communities as a source of AK:

- *(B-1) Diversity of solutions and opinions*: The amount of discussions and opinions provide architects with an initial list of architectural solutions, which are candidates for experimentation and prototyping. This diversity of opinions is an advantage of developers communities

over seeking knowledge from a single expert, who might not have a broader knowledge but only specific knowledge about a problem or technology.

- *(B-2) Solution evaluation*: Asking a question in a developer community about a particular architectural solution, and asking to assess the solution for a certain problem is a common practice mentioned by participants. In addition, developer communities provide comparisons between technology solutions regarding their quality attributes (e.g. performance benchmarks).
- *(B-3) Communication between architects and technologists*: Developer communities act as a "social network" for architects and experts from technology vendors. For example, interviewee I1 mentioned that "*Sometimes, there is not enough information available, and you cannot find what you want, but you can contact well-known experts in developer community directly, specially experts who work in the company, which develop technology products*". This communication is important especially when working with new technologies that have not yet been sufficiently documented or explored by other users.
- *(B-4) Solutions for a single and focused design problem*: Each discussion in the developer community focuses on a single problem and its varied solutions (see also B-1). This is an advantage over trying to identify reusable knowledge in complete architectural documentations of previous systems, which provide a complete solution for several design issues.
- *(B-5) No financial costs*: AK in developer communities is free of charge. This is different than asking help from an expert (e.g. technology consultants), who might charge their clients for sharing their knowledge based. For example, interviewee I8 mentioned that "*We need to get all possible knowledge from the communities before asking an expert to minimize the costs of consultancy*".

6.3.2 Problems Faced when Searching for AK in Developer Communities

We identified the following problems that architects face when searching for AK in developer communities:

- *(P-1) Complexity to describe design problems*: It is complex to describe a design problem for which a solution is sought. As interviewee I6 mentioned, " *It is hard to find the same design situation in developers community. The problem is in the way to describe the problem and to reach it in the discussions, this connection is not there. (...) In programming problems, you can search with the exception code and description, while design problems have no code or clear description*".
- *(P-2) Complex semantics of discussions*: After posting a question and retrieving a list of the most relevant posts from communities, practitioners need to read carefully the discussions to understand the problems and their context. A careful understanding of text is essential, because even similar design problems often involve lots of contextual differences. One reason behind the textual complexity is the incomplete information provided by the users who cannot share the full design context (e.g. business goals). Users try to overcome this problem (lack of full context) through writing all possibilities using conditional statements, which makes the text lengthy and obscure.
- *(P-3) Required domain knowledge*: When the architect is not an expert in the problem domain, it becomes harder to specify useful and meaningful keywords for a search in developer communities. This problem is driven by the fact that developer community websites

are scattered among different technological communities and not categorized according to the domain or types of architecture design problems. Moreover, understanding the discussions in developer communities requires familiarity with terms and concepts in this domain. This makes searching based on keywords searching even harder.

- *(P-4) Lack of trust in community*: Some of the information in developer communities could be wrong and misleading. Thus, architects need to be cautious in considering the proposed solutions and their evaluations. When depending on untrusted sources of AK, it is also harder to agree on a decision within a group [RM14]. As interviewee I6 stated, "*Even if the best solution is written in forums, the forum needs to be trusted and convince people that this is the best practice in order to reduce conflict*".

- *(P-5) Varying contexts for similar problems*: Even though communities provide large amounts of knowledge, it is hard to find an exact match for a given problem. This is because complex design problems involve different possibilities of components design, technologies, and requirements. This is different to programming problems, which are very specific. Interviewee I2 stated that "*In programming, you can find a snippet code, but in architecture, you need to understand every word (...) because all what you read could be different from your design situation*". Communities enhance the knowledge of architects to solve problems, but do not provide a packaged final solution.

- *(P-6) Limited capabilities of search engines for keyword-based searches*: There are fewer architecture-relevant discussions in developer communities compared to discussions around programming problems. This makes it challenging for keyword-based searches to find relevant AK in developer communities.

6.3.3 Solutions to Improve the Search for AK

Interviewees proposed the following solutions to improve AK sharing and reuse:

- *(S-1)"Architecture-aware" semantics*: Instead of depending on keywords to create a search query for AK on developer communities (problem P-1), architects would benefit from describing queries using architecture concepts. For example, a query could support describing a component design of a system using different levels of abstraction. Interviewee I5 suggested that "*I would like to specify the system applications and their technologies, and write in a text to define my question, and I would like to get all similar issues, and proposed solutions, with similar situations but different technologies*".

- *(S-2) Semantic evaluation for the validity of opinions*: Information validity is a problem (problem P-4). As interviewee I1 stated, "*It would be good to differentiate between opinions, and already experimented and tried solutions*". Thus, additional possibilities for evaluating the textual content of communities is needed. Interviewee I3 proposed that "*A community of experts should review the discussion to assess their relevance and validity*".

- *(S-3) Knowledge categorization*: Discussions in communities are usually categorized based on the used technologies [TBS11a]. If we categorized discussions differently from an architecture perspective (e.g. based on quality attributes, types of design problems), it could facilitate searching for architectural problems and solutions to overcome the drawbacks of keywords search engines (problem P-6). For example, interviewee I9 stated that "*We need a better sorting and categorization of discussions according to the design problems, the same design could be linked to different technologies and industries*".

Table 6.2: Benefits, problems and solutions mapped to interviewees

	Interviewee									
	I1	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**
Benefits										
B-1	x	x	x		x	x		x	x	x
B-2	x		x	x	x				x	
B-3	x						x			
B-4	x							x		
B-5							x	x		
Problems										
P-1				x	x	x		x		x
P-2	x	x	x	x				x		
P-3		x		x				x		x
P-4	x	x				x			x	
P-5	x	x			x	x		x	x	x
P-6				x	x	x		x	x	
Solutions										
S-1		x			x	x				x
S-2	x		x					x		x
S-3			x		x			x	x	
S-4				x					x	
S-5		x		x						
S-6	x		x		x		x	x		x

- *(S-4) Associate other AK resources to discussions*: To support better understanding in discussions, AK might be linked to other sources to better understand the context of discussions and solutions. For example, interviewee I4 proposed to "*Provide references for resources (e.g. papers or books) to understand the domain and the used notation*". This would help architects, who are unfamiliar with a certain domain (problem P-3). This aspect is important in relating the AK in developer communities with existing AK repositories.

- *(S-5) Offer recommendations during design*: AK should be available during design activities, in real-time and without much effort to describe design problems (P-1) and to search for solutions (P-6). Interviewee I2 suggested to "*... provide design recommendations during work depending on the architectural situation, for example to know other alternatives for a technology, or better components and technology integration*". Thus, information about context need to be captured (automatically or semi-automatically).

- *(S-6) Unified social network for AK*: Current software architecture social networks spread across technology providers in different domains (e.g. as communities of practice). This complicates searching for AK using search engines (problem P-6). Interviewees proposed creating a social network for software architects across different technology products and domains. Such social network should a) aggregate distributed architecture-relevant discussions from different communities, and b) help communicate architecture problems and solutions (e.g. sharing models).

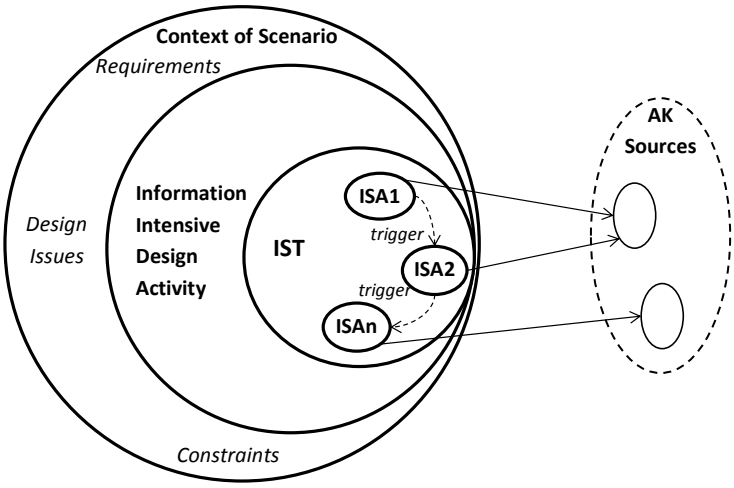


Figure 6.1: The main elements of a scenario to search for architecture knowledge.

6.4 Scenarios for Searching AK in Developer Communities

This section presents two contributions. First, we present in Section 6.4.1 the different conceptual elements of a scenario to search for architectural information. Second, we present in Section 6.4.2 different scenarios and use-cases to search for architectural information in developer communities. The contributions in this section provide an answer to RQ7.

6.4.1 Conceptual Elements of Scenarios to Search for AK

Architecture design activities are conducted within a design context, which involves requirements, constraints and design issues. Some design activities require significant amounts of information from sources of AK. We call these design activities *information-intensive design activities*. For example, architecture synthesis [HKN+07] is an information-intensive design activity, because it requires searching for architectural solutions. On the other hand, documenting the architecture might not involve significant information seeking tasks, but other tasks (modeling, technical writing, etc.). Therefore, documenting the architecture is not an information-intensive design activity. During each information-intensive design activity, architects go through scenarios to search for AK.

Fig. 6.1 shows an illustrative diagram for the main elements of a scenario to search for AK. Each scenario starts from an information intensive design activity, which involves one or more *information seeking task* (IST). An IST is a task to search and gather specific information from one or more sources of AK. The gathered information supports the architect to perform and complete the rest of the tasks in the design activity. In order to complete ISTs, architects need to decide on the possible sources of AK. Architects could rely solely on their own knowledge [HA11], or seek help from external sources of AK (e.g. communities). To request assistance from external sources of AK, architect performs several *information searching activities* (ISAs). For example, "Compare ActiveMQ and RabbitMQ regarding their performance" is an IST, which needs to be completed to perform the "solution synthesis" design activity. To complete this IST, architects could perform the following ISAs: ask colleagues in the company about their experience with both technologies, or search in online communities using search engines. Executing ISAs depends on the design context (i.e. requirements, constraints, design issue), and the goal of the IST, which initiated the

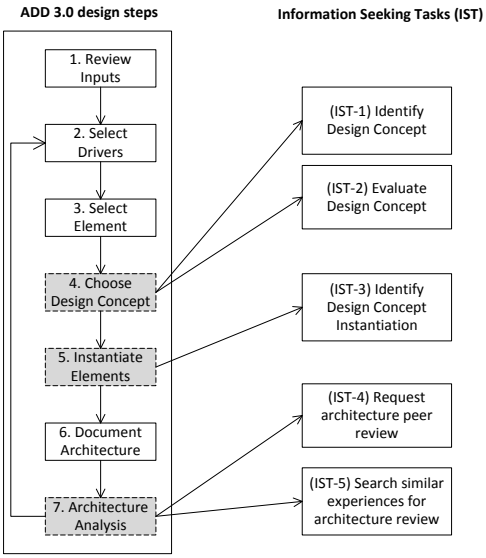


Figure 6.2: Relating information intensive architecture design activities (in gray) with ISTs

searching scenario. For example, when searching in online communities using keyword-based search engines, architects could use keywords which are derived from the design context (e.g. name of technology, domain name, type of design problem) and the IST (e.g. keywords like "difference" or "comparison" for ISTs, which compare and evaluate technologies).

Searching for AK can be considered an iterative learning process. If an ISA does not provide sufficient knowledge to complete an IST, the architect would continue performing new ISAs until sufficient knowledge has been gained to confidently complete an IST (within time and resource constraints). This iterative learning characteristic distinguishes searching for AK from searching for programming problems, where specific and final programming solutions (e.g. code snippets) could be found (see P-5 in Sec. 6.3.2).

The conceptual elements in Fig. 6.1 provide the template for architectural design information retrieval tasks. Based on this model, we created tasks, which have been used to experiment our proposed searching approach in Chapter 8.

6.4.2 Examples and Types of Scenarios for Searching AK in Developer Communities

In this section, we present ISTs and ISAs identified in our study. We also present examples of analyzed scenarios (obtained from the interviews) which combine concrete ISTs and ISAs.

To identify ISTs and ISAs, we followed several steps:

1. We first analyzed which ADD-3.0 steps are information-intensive design activities. As mentioned earlier in Sec. 6.2.2, we focused on ADD-3.0 as an exemplar architecture design process [KC16]. This step resulted in three information-intensive design activities (left side of Fig. 6.2).

2. Then, for each information-intensive design activity in ADD-3.0, we determined ISTs that need to be performed in order to complete that information-intensive design activity. This

Table 6.3: Information searching activities (ISAs) for AK in developer communities

ID	Information Searching Activity (ISA)
ISA-1	Search community through keywords search engines
ISA-2	Ask an expert in the community for specific information
ISA-3	Identify and browse architecture experts in communities
ISA-4	Browse and filter individual statements that contain AK in community pages
ISA-5	Evaluate the validity of opinions in community pages
ISA-6	Identify architecture-relevant community pages
ISA-7	Write and submit a question to community

step resulted in five ISTs. Fig. 6.2 links the three information-intensive design activities in ADD-3.0 (highlighted in grey) to their ISTs.

3. To get a list of ISAs, we annotated potential ISAs in scenarios and analyzed all scenarios mentioned by the interviewees. This resulted in a list of ISAs as presented in Table 6.3.

Below we explain the links between ADD-3.0 design activities, ISTs and ISAs using examples of scenarios obtained in the interviews:

- Architects start the architecture design process with three analysis steps (as defined in ADD-3.0): *1. Review inputs* (e.g. requirements, constraints), *2. Select drivers* (e.g. quality attributes) and *3. Select element* from existing designs (if any).
- Next, architects conduct two synthesis steps which are both information-intensive architecture design activities:

  4. Choose design concept involves two ISTs: "Identify design concept" **(IST-1)** and "Select design concept" **(IST-2)**. In both ISTs, architects need to search for sources of AK (e.g. experts, books, developer communities), and collect information about architectural solutions (e.g. alternative technologies), including their evaluation regarding quality attributes. For instance, interviewee I8 mentioned the following scenario:

  "*We were searching for an n-memory grid, with replication and high availability* **(IST-1)**. *We search in Internet using search engines, there were much options* **(ISA-1)** *However, the customer forced us to use a certain technology from a vendor, we were not familiar with this technology, so we had to evaluate if it fits to the requirements* **(IST-2)** *We searched again about this technology* **(ISA-1)** *we found lots of relevant information, we analyzed them* **(ISA-4)** *but they were not the same scenario as our requirements, through the communities, we identified an expert, who is one of the developers of the product* **(ISA-3)** *we consulted him, and he guide us to the right way of using the technology in our scenario* **(ISA-2)**."

  5. Instantiate Elements involves one IST: "Identify design concept instantiation" **(IST-3)**. Here, the architect tries to determine the most suitable components configuration for the previously selected architectural solution (e.g. number of layers after selecting the layers pattern). For instance, interviewee I6 mentioned the following scenario:

  "We had a ticket system and a CRM, which communicate with each other through an integration layer, both systems use different formats of data, so our design issue was to determine, which components should convert the data **(IST-3)** *We searched for best practices in Telecom forums* **(ISA-1)** *We select forums, which are trusted and respected by all partners"* **(ISA-5)**

- After determining the solution, architects document the proposed architecture (ADD-3.0 step 6).
- In step *7. Architecture analysis*, several architecture analysis methods could be used. Some of them involve ISTs. For example, to conduct an architecture peer review **(IST-4)**, one needs to identify experts who can review the architecture **(ISA-4)**. Another method for architecture analysis is searching for similar experiences **(IST-5)**. The architect might consider several sources of AK (e.g. existing architecture documents) or seek help from an expert, who solved the same problem before **(ISA-2, 4)**.

The analysis of searching scenarios support the creation of searching tasks, which have been used to evaluate our approach to search for architectural information in developer communities (see Chapter 8).

6.5 Discussion

6.5.1 Interpretation of Results

The findings presented in this chapter provide insights about the current state of practice of using AK from developer communities. Developer communities are traditionally used to solve programming problems (and were therefore researched in the past from a developer's rather than from an architect's point of view). Our study is the first that *empirically* analyzes and describes the use of developer communities for solving architecture-related design problems.

The problems mentioned by practitioners when using developer communities for architectural design show and verify the complexity of searching for relevant architectural information, when solving a design issue. Our results align and verify with other experiences in the current state of the art. For example, Gorton et al. experienced complexity to find relevant architectural information when using commercial search engines. Our conceptual model to search for architectural information as well as our identified use-cases show the iterative nature of searching for architectural information. Software engineers seek information from one source of knowledge, which could motivates another search in another source of knowledge. Moreover, the use-cases show some of the different stations, where software engineers go through to search for architectural information. Therefore, approaches for architecture knowledge management need to consider this iterative nature of searching, as well as the different use-cases to find architectural information.

6.5.2 Implications of Results on further Research

The identified benefits, problems, and solutions mentioned by practitioners (Sec. 6.3) have the following implications on further research steps:

1. *Tools to search for AK in communities*: Most of the problems and solutions mentioned by the interviewees propose creating tools to facilitate using AK in communities. Relationships between problems and solutions could suggest certain approaches. For example, problems P-1 and P-6 and solution S-1 suggest building specialized semantic search engines (e.g. [FCL+11]) to search for AK in communities.

2. *Developing a new unified social network for software architecture*: Solution S-6 is the most common solution among the interview participants. A specialized social network for software architecture would not only facilitate AK sharing, but it will increase the awareness about the importance of software architecture among other software communities. The interviewees mentioned some of the obstacles and solutions for building such a social network. Further research efforts need to identify additional obstacles and solutions.

With regards to tool development, answers to RQ7 provide empirically-grounded requirements and scenarios. For instance, our discovered ISTs and ISAs support defining requirements and use cases for tools. We present below two possible examples for requirements in an AK management system based on ISTs and ISAs:

1. Based on IST-3 and ISA-2, a requirement could be "*The system should provide the possibility for architects to ask experts for suggestions on components design of the system*".
2. Based on IST-1, ISA-4, and ISA-6, a requirement could be "*The system should provide the ability to search for suitable and reliable design concepts through browsing the personal profile of experts in relevant domains*".

Furthermore, ISTs and ISAs could be used to build frameworks to evaluate information retrieval tools for AK searching. The evaluation of information retrieval tools is usually done through simulated tasks [Bor00], which need to simulate real scenarios. The list of identified ISTs and ISAs provides a reference list for possible types of tasks, which need to be considered to evaluate tools.

The results in this chapter motivated and supported our work in this thesis to develop an effective approach to search for architectural information in developer communities. In Chapter 8, we used our proposed conceptual model in Section 6.4.1 to develop a template for an architectural searching task. This template has been used during the evaluation of our proposed search approach. We additionally used the scenarios provided by practitioners in this chapter to create searching tasks to evaluate our proposed enhanced search approach.

7

Classification Approaches for Architecture-relevant Posts

7.1 Research Questions and Contributions 114
7.2 Corpus to Evaluate Classification Approaches 115
7.3 Exploring Tags for Identifying Architecture-relevant Posts 116
7.4 Classifying Stack Overflow Posts for Architectural Relevance 117
7.5 Evaluation of Classification Approaches 121
7.6 Significant Terms and Ontology Classes 127
7.7 Discussion 131

7.1 Research Questions and Contributions

This chapter provides answers for **RQ8** and **RQ9**:
RQ8: How can we automatically identify architecture-relevant posts in Stack Overflow and separate them from pure programming-related posts?

RQ9: What are terms and ontology concepts in Stack Overflow posts, that distinguish ARPs from programming posts?

By answering RQ8 and RQ9, we support achieving the third goal of the dissertation "Propose approaches to search for architectural information in developer communities" (see Chapter 1). To answer RQ8 and RQ9, we develop and compare a set of classification approaches to automatically identify and classify architecture-relevant posts in online developer communities. In detail, our **contributions** are the following:

- We develop several classification approaches (Section 7.4) to automatically separate architecture-relevant posts from pure programming-related posts in Stack Overflow. The classification approaches also classify architecture-relevant posts into sub-categories to support specific architecture design steps. The classification approaches are combinations of different ways of preprocessing posts (i.e. transforming the text of posts into feature vectors for machine learning algorithms) and existing machine learning-based classification algorithms. Some of the approaches not only consider the textual content of posts, but also use our proposed AK ontology (see Chapter 5), which captures concepts relevant to software architecture.

- To evaluate and compare the classification approaches and to decide which combination of preprocessing approach and classification algorithm works best (Section 7.5), we conducted experiments using our proposed corpus of classified Stack Overflow posts (see Chapter 4).
- We identify in Section 7.6 the most significant terms and ontology classes to identify and classify ARPs in developer communities. The terms and ontology classes are sorted according to their impact on classification.

The classification approach is a first step towards automatically identifying AK in online communities based on semantic information captured in an ontology of AK concepts rather than depending solely on lexical features (i.e. words in text). Also, it is the first approach that utilizes online developer communities to support architecting tasks and extends recent efforts to capture AK from textual contents of web pages [GXY+17] and issue trackers [BSB+17]. Moreover, our proposed classification approaches support developing an enhanced search approach for architectural information in developer communities answering (see Chapter 8).

Before presenting our classification approaches, and their evaluations in Sections 7.4 and 7.5. We explore first the possibility of using using tags assigned by community users to identify ARPs in Section 7.3. We show that ARPs cannot easily be identified using the tagging system offered by many online developers communities. Therefore, we propose text-based classification approaches in Section 7.4 which we then evaluate in Section 7.5. In Section 7.6, we present the most significant terms and ontology classes to identify and classify ARPs. Finally, we discuss the results of our work in Section 7.7.

7.2 Corpus to Evaluate Classification Approaches

The classification approaches presented in this chapter rely on supervised classification algorithms. To develop, analyze and test the classification approaches and to train classification algorithms, manually classified posts are required. In this section, we describe our corpus, which is used for training and evaluating the classification approaches. The corpus consists of two samples:

1. *Development Sample*: This sample is used to evaluate the accuracy of the classification approaches. In addition, it is used to answer RQ9, and support analyzing the characteristics of the classification approaches.
2. *Testing Sample*: This sample is used only for testing the proposed classification approaches regarding its generalizability of our approach.

Both samples are presented in the following sub-sections.

7.2.1 Development Sample

The development sample used our existing corpus of Stack Overflow posts (see Chapter 4). These posts were classified by practitioners as architecture-relevant posts (including the type of ARP) or programming posts. The development sample includes two datasets:

1. *2-class dataset*: This dataset includes two classes of posts (858 ARPs, 1,653 programming posts). We used this dataset to test how developed classification approaches separate ARPs from programming posts.

2. *4-class dataset*: This dataset includes four classes of posts (1,653 programming posts and three compact types of ARPs [282 technology identification posts, 291 technology evaluation posts and 285 features and configuration posts]). Appendix A provides list of ARPs in the development sample. Moreover, we provide complete list of posts online[1] and in the companion CD.

7.2.2 Testing Sample

The data of the testing sample has been collected and evaluated using the same methodology as our corpus (see Chapter 4), but instead of using middleware posts we collected posts related to big-data. In addition, the data has been classified similar to the development sample (i.e. 2-class and 4-class datasets):

1. *2-class dataset*: This dataset includes two classes of posts (172 ARPs, 143 programming posts).

2. *4-class dataset*: This dataset includes four classes of posts (143 programming posts and three types of ARPs [50 technology identification posts, 65 technology evaluation posts and 57 features and configuration posts])

In summary, we followed these steps to gather the testing sample:

1. Gather Stack Overflow posts, which discuss questions related to bigdata technologies. We chose big-data as the "topic" of our corpus since it is an established and recent topic in software architecture (e.g. [GKN15]). Posts were collected using SQL queries through the StackExchange Explorer based on keywords of technology names. See Appendix A regarding the names of big-data technologies considered in queries.

2. Classify Stack Overflow posts manually into the three types of ARPs and programming posts using a summarizing qualitative content analysis method [May14].

3. Evaluate classification of posts through feedback from practitioners. 50% of the posts have been given to practitioners to verify their architecture relevance. The evaluation resulted in 87% agreement.

7.3 Exploring Tags for Identifying Architecture-relevant Posts

Some online developer communities (including Stack Overflow) allow users to tag posts to specify the topic of a post. Therefore, we explored if tags are sufficient to distinguish ARPs from programming posts. We did this using the 2-class dataset in the development sample, which is discussed in Section 7.2. For each tag used for posts in that dataset, we calculated the percentage of ARPs and programming posts that use that tag (relative to the total number of ARPs and programming

[1] https://swk-www.informatik.uni-hamburg.de/~soliman/Dissertation.zip

Table 7.1: Top tags with highest differences in occurrences between Stack Overflow programming posts and ARPs

Tag	% of ARPs	% of programming posts	Difference in %
web-services	11.4%	4.92%	6.47%
soap	9.19%	3.12%	6.07%
message-queue	6.05%	0.9%	5.14%
rest	6.86%	2.01%	4.85%
java	15.81%	11.16%	4.66%
rabbitmq	8.37%	4.12%	4.25%
jms	6.05%	1.91%	4.14%
zeromq	4.3%	0.5%	3.8%
rpc	4.4%	0.8%	3.61%
soa	3.7%	0.3%	3.42%
architecture	3.49%	0.1%	3.39%

posts). We then compared these percentages between ARPs and programming posts to see if there are tags that appear differently in both types of posts. Table 7.1 shows the top list of tags with the biggest differences in their occurrences between ARPs and programming posts. Further details and results of the analysis can be found in Appendix C and in the companion CD.

Furthermore, we tested if there is a significant difference in the occurrences of tags between both types of posts using a t-test with equal variances [Mon06]. We found that there is no statistical significant difference ($t(2488) = 0.464$, $p = 0.642$). Table 7.1 shows that the biggest difference in the occurrence of tags is for the tag "web-services". However, this difference is only 6.47% (i.e., the "web-services" tag appears 6.47% more frequently in ARPs than in programming posts). Moreover, seven tags out of the top eleven tags are technology names. The "architecture" tag only appears 3.39% more frequently in ARPs than in programming posts and more than 96% of ARPs have no "architecture" tag. Therefore, we conclude that tags are not sufficient to differentiate between architecture and programming posts. Instead, we need to examine the content of posts using text-based classification, which is presented in the next section.

7.4 Classifying Stack Overflow Posts for Architectural Relevance

7.4.1 Overview

The goal of our text-based classification is to classify posts into *predefined* categories (three types of ARPs and programming posts, see Section 7.2). This is a *text classification* problem. Therefore, topic modeling approaches (e.g. LDA) to discover undefined topics are not applicable. The classification approaches that we developed have two main components:

1. *Preprocessing*: Preprocessing transforms words in posts into a feature vector. A feature vector is a vector of numbers , which represents important classification features. Natural language has a lexical and a semantic dimension [Gle05]. Therefore, we explored two preprocessing approaches. Each approach produces different *classification features*:

   (a) *Lexical* features of text in posts using the *Bag-of-Words* approach [MS99] (Section 7.4.2).

(b) *Semantic* features of text in posts using an *ontology-based* classification which captures architecture-relevant semantics [FGWY07] (Section 7.4.3).

2. *Classification algorithm*: Classification algorithms use feature vectors to develop a classification model to automatically categorize posts. The model is developed through training previously classified posts, and then used to classify new posts. We propose the use of different classification algorithms (e.g. Naive Bayes, random forest) in Sections 7.4.2, 7.4.3 and compare them in Section 7.5.

We also experimented with an *ensemble learning* approach [Rok10] to combine the results of different classification approaches into a single approach (Section 7.4.4).

7.4.2 Bag-of-Words Classification

Bag-of-Words uses individual words in posts to determine characteristics of the text [SM86]. The frequency of each word as well as the frequency of sequences of words in posts are used as features for training classification algorithms. An advantage of this approach is its simplicity and ability to capture lexical differences in words [SM86].

Preprocessing: To transform textual posts into a feature vector, we first removed stop words from posts[1]. This reduces noise and increases the chance for distinctive sequence of words to be transformed into single features. Then, we used an n-gram sequence classification approach [XPK10] to capture the sequence of common words. The frequency of each unique sequence of words is treated as a feature.

Classification algorithms: We used the feature vectors with the Naive Bayes [JL95] and Bayesian Network [FGG97] algorithms. These algorithms are known for their ability to classify documents based on sequences of textual segments [JL95, FGG97]. We used Weka [HFH+09] to experiment with variations of n-gram sequences: n= 1, 3 and 5 for Naive Bayes (VBnG), and n = 3 and 5 for Bayesian Network (BNnG). We found that any $n > 5$ did not yield better results. The evaluation results for the different algorithms are presented in Section 7.5.

7.4.3 Ontology-Based Classification

The classification approaches developed in this section use an ontology-based document classification approach [FGWY07]. Our proposed AK ontology in developer communities (see Chapter 5) has been used to implement the proposed classification approaches in this sub-section. To explore both the impact of individual ontology classes and sequences of ontology classes on identifying architecture-relevant posts, we separately explored single-ontology-class and multi-ontology-class classification approaches.

[1] Stop words are the most commonly used words in a language, For example, prepositions (“in”, “at”, etc.), pronouns (“he”, “it”, etc.), and articles (“the”, “a”). We used a standard list of stop-words http://astellar.com/2011/12/stopwords-for-sphinx-search/

7.4.3.1 Single-ontology-class Classification

The single-ontology-class approach relies on features from separate simple ontology classes and lexical triggers (see Chapter 5).

Preprocessing: We used two types of features, which were derived from simple ontology classes and lexical triggers.

1. *Total number of instances*: Total number of words in a post which belong to a simple ontology class or lexical trigger (we use the prefix "num" before the ID of the ontology class to refer to these features).
2. *Distinct number of instances*: Distinct number of words in a post (without duplicates) which belong to an ontology class (we use the prefix "dist" before the ID of the ontology class to refer to these features). This type of feature is used to distinguish posts which have multiple different words from the same ontology class. For example, posts that discuss different technologies and quality attributes might be about evaluating technologies, since they compare architectural alternatives rather than details about implementing one particular software functionality using a specific technology.

An example of the use of the two types of features is the following: If "WCF" was mentioned three times in a post, and "RabbitMQ" twice, the feature *numTEC* will be counted five times, while the feature *distTEC* will be counted twice for ontology class *Technology Solution (TEC)*.

We also consider some features which are commonly used (e.g. [ZYL+15]) to classify Stack Overflow posts:

1. *hasSourceCode*: Boolean feature which checks whether an answer part of a post contains source code. Source code is a characteristic of programming posts [NSMB12b] and therefore hints to posts which potentially are not architecture-relevant.
2. *numParagraphs*: Numerical feature that counts the number of paragraphs in a post [ZYL+15]. Architecture-relevant posts may be longer than programming posts because they include more detailed discussions about the context of a problem, design alternatives and their implications.
3. *numWords*: Similar to numParagraphs, but counting the number of words in a post [ZYL+15].

We used the frequency of terms for weighting our feature, after comparing its results with the term frequency-inverse document frequency [WFH11].

Classification algorithms: We used Weka [HFH+09] to compare five algorithms: decision trees (J48) [Qui93], random forest (RF) [Bre01], decision table (DT) [Koh95], support vector machines using the sequential minimal optimization (SVM) [Pla99], and logistic model trees (LMT) [LHF05]. These algorithms were chosen because they have previously been used successfully for document classification [Seb02] and software engineering problems [TR16].

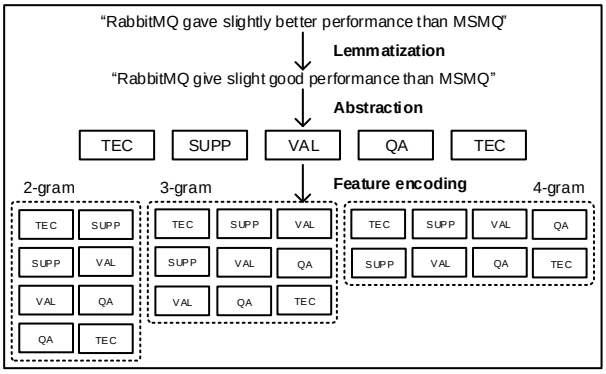


Figure 7.1: Example of multi-ontology-class preprocessing

7.4.3.2 Multi-ontology-class Classification

The single-ontology-class classification does not identify composite ontology classes. Therefore, we also explore multi-ontology-class classification, which treats each possible sequence of single ontology classes and lexical triggers as features.

Preprocessing: To transform text in posts into features (i.e., sequences of ontology classes), we followed the steps below (Fig. 7.1 shows an example):

1. *Lemmatization* converts the inflected forms of a word into a single form so they can be analyzed as a single item. This allows capturing more words for one ontology class. We used the Stanford Core NLP[1] lemmatizer.

2. *Abstraction* replaces all words which belong to one of the ontology classes with the name of that ontology class. Words which do not belong to any of the ontology classes are removed. This step results in a sequence of ontology class names for each post.

3. *Sequential feature encoding* captures all possible combinations of sequences of ontology classes in a post through *n*-gram processing. For example, by applying 5-gram on the identified ontology classes for the sentence in Fig. 7.1, we get a total of 15 sequences between 1-gram and 5-gram. The frequency of each unique sequence of ontology classes is treated as a feature.

Classification algorithms: We processed feature vectors with the Naive Bayes [JL95] and Bayesian Network [FGG97] algorithms since these algorithms are suitable for classifying documents based on sequences of textual segments [JL95, FGG97]. We used Weka [HFH+09] to experiment with variations of *n*-gram sequences [XPK10]: *n*= 1, 3 and 5 for Naive Bayes (VB*n*G), and *n* = 3 and 5 for Bayesian Network (BN*n*G). These are the same classification algorithms as used with the Bag-of-Words, but applied to sequences of ontology classes rather than to sequences of words.

7.4.4 Ensemble Learning

In the previous sections, we introduced different classification approaches, which use three preprocessing techniques to define features and feature vectors (Bag-of-Words, single-ontology-class,

[1] http://stanfordnlp.github.io/CoreNLP/

and multi-ontology-class) and different classification algorithms. Combining the different approaches together would make use of the benefits of each to potentially improve the quality of classification. To achieve this, we performed ensemble learning using the *voting algorithm* [KHDM98]. The voting algorithm takes the probabilities of a post for being of a certain type as computed by each classification approach and calculates an average probability. We chose the voting algorithm for ensemble learning after a comparison with the *boosting ensemble learning algorithm* [Kun04]. The boosting ensemble learning algorithm did not improve the quality of the classification above the quality of the individual classification approaches combined in the ensemble.

7.5 Evaluation of Classification Approaches

The comparison of the classification approaches from Section 7.4 aims at identifying approaches which are most accurate for identifying and classifying architecture-relevant posts. We evaluated classification approaches regarding two main aspects:

1. *Accuracy of classification approaches*: The accuracy to identify and classify ARPs from the same domain. The evaluation uses the development sample from our evaluation corpus in Section 7.2.1. The results of the evaluation is presented in sub-section 7.5.1.
2. *Generalisability of classification approaches*: Their accuracy to identify and classify ARPs from a different domain. The evaluation uses the testing sample from our evaluation corpus in Section 7.2.2. The results of the evaluation is presented in sub-section 7.5.2.

7.5.1 Accuracy of Classification Approaches

7.5.1.1 Study Design

To train and test the different classification approaches, we used both the 2-class and 4-class datasets from the development sample as described in Section 7.2.1. The 2-class dataset was used to classify posts into ARPs and programming posts. The 4-class dataset was used to classify ARPs into sub-types. To compare the different classification approaches, we performed a ten-fold cross-validation [Bis06]. All posts in the corpus were randomly assigned to ten groups. We then trained the different classification algorithms on nine groups and tested them on the remaining group. Each group served as testing group once, i.e., we performed ten experiments. We calculated true-positives (*TP*), true-negatives (*TN*), false-positives (*FP*), and false-negatives (*FN*). We then computed for each of the ten experiments the precision *P*, recall *R*, and F-score *F1* as the harmonic mean of precision and recall. The larger *F1*, the "better" the results.

7.5.1.2 Results

Tables 7.2, 7.3 and 7.4 compare the different classification approaches using the different methods of preprocessing: Bag-of-Words, single-ontology-class and multi-ontology class classification respectively. The comparison is based on the average *P*, *R* and *F1* across the ten experiments for each classification approach. We highlight the best performing classification algorithms in bold. The *best performing classification approaches* A1 to A3 are:

Table 7.2: Comparison of Classification Approaches using the Bag-of-Words

		Classifiers used with Bag-of-Words				
Data-set		VB1G	VB3G	VB5G	BN3G	BN5G
2-Classes	***P***	0.777	0.778	0.781	0.769	0.770
	R	0.774	0.780	0.783	0.771	0.772
	F1	0.775	0.779	**0.781**	0.770	0.771
4-Classes	***P***	0.729	0.732	0.732	0.726	0.730
	R	0.686	0.698	0.697	0.711	0.715
	F1	0.703	0.712	0.711	0.718	**0.722**

Table 7.3: Comparison of Classification Approaches using the Single-Ontology-Class

		Classification Approaches using Single-Ontology				
Data-set		J48	RF	DT	SVM-SMO	LMT
2-Classes	***P***	0.736	0.811	0.760	0.813	0.817
	R	0.739	0.812	0.765	0.816	0.819
	F1	0.737	0.806	0.756	0.811	**0.816**
4-Classes	***P***	0.633	0.677	0.656	0.694	0.725
	R	0.656	0.740	0.715	0.743	0.756
	F1	0.644	0.677	0.656	0.694	**0.725**

Table 7.4: Comparison of Classification Approaches using the Multi-Ontology-Class

		Multi-Ontology-Class Classification Approaches				
Data-set		VB1G	VB3G	VB5G	BN3G	BN5G
2-Classes	***P***	0.718	0.743	0.766	0.745	0.745
	R	0.703	0.729	0.767	0.742	0.742
	F1	0.708	0.733	**0.766**	0.743	0.743
4-Classes	***P***	0.660	0.695	0.701	0.686	0.689
	R	0.635	0.626	0.652	0.645	0.65
	F1	0.630	0.651	**0.672**	0.661	0.666

Table 7.5: Comparison of the Best Performing Classification Approaches and their Combination using the Ensemble Learning

Pre-processing		**Bag-of-Words**	**Single-Ontology Class**	**Multi-Ontology Class**	**Ensemble**
Classification Algorithms		**VB5G**	**LMT**	**VB5G**	**Voting**
2-Classes	***P***	0.781	0.817	0.766	0.847
	R	0.783	0.819	0.767	0.840
	F1	0.781	0.816	0.766	**0.842**
Classification Algorithms		**BN5G**	**LMT**	**VB5G**	**Voting**
4-Classes	***P***	0.730	0.725	0.701	0.737
	R	0.715	0.756	0.652	0.733
	F1	0.722	0.725	0.672	**0.734**

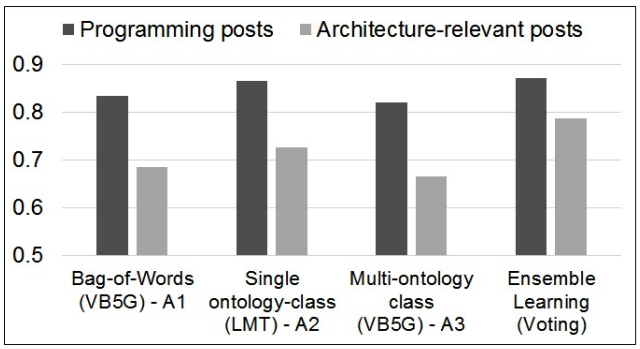


Figure 7.2: Comparing F-scores across different classification approaches and types of posts in the 2-class dataset

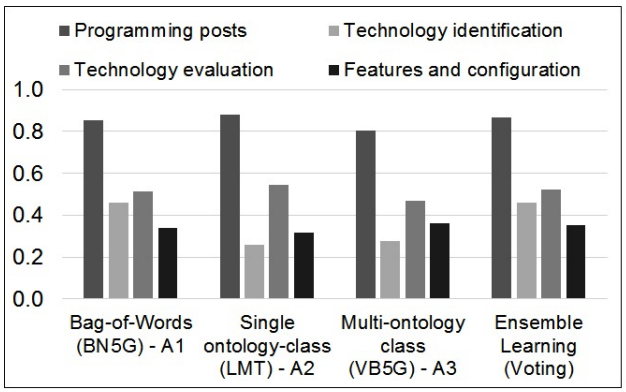


Figure 7.3: Comparing F-scores across different classification approaches and types of posts in the 4-class dataset

- **A1:** Bag-of-words using algorithms VB5G for 2-class dataset and BN5G for 4-class dataset.
- **A2:** Single-ontology-class classification using LMT (for both data sets).
- **A3:** Multi-ontology-class classification using VB5G (for both data sets).

The best classification result is achieved by the ensemble learning using the voting algorithm when it combines the results of the best performing classification algorithms A1, A2, and A3. Table 7.5 shows the results of the voting ensemble learning.

To further understand the results in Tables 7.2, 7.3, 7.4, and 7.5, and to understand which classification approaches work better for which type of posts (ARPs and programming posts), we calculated *F1* for each type of posts using classification approaches A1, A2 and A3. Fig. 7.2 and 7.3 show a comparison of the classification approaches for both the 2-class and 4-class datasets. From Fig. 7.2, we make two observations:

1. All classification approaches are better at identifying programming posts than at identifying ARPs.
2. The single-ontology class classification (using LMT) performs better for identifying ARPs compared to the multi-ontology class classification (using VB5G) and Bag-of-Words (using VB5G).

Based on Fig. 7.3, we observe the following:

Table 7.6: Confusion matrix for classification approaches

		True Programming	True Technology identification	True Technology evaluation	True Features and configuration
A1: Bag of Words (VB5G)	*Programming*	**1,354**	47	53	92
	Technology identification	53	**82**	24	21
	Technology evaluation	77	35	**163**	72
	Features and configuration	141	14	47	**100**
A2: Single ontology class (LMT)	*Programming*	**1,546**	102	84	154
	Technology identification	20	**33**	18	8
	Technology evaluation	24	27	**146**	53
	Features and configuration	35	16	39	**70**
A3: Multi ontology class (VB5G)	*Programming*	**1,224**	59	41	90
	Technology identification	104	**58**	59	22
	Technology evaluation	86	46	**144**	51
	Features and configuration	211	15	43	**122**

1. The Bag-of-Words approach (using BN5G) achieves a higher *F1* than the ontology-based approaches when classifying "Technology identification" ARPs.
2. The single-ontology-class approach (using LMT) yields better results when identifying programming posts than any of the other approaches. On the other hand, it performs poorly when identifying "Technology identification" posts.
3. The multi-ontology-class classification (using VB5G) is better at identifying "Features and configuration" ARPs than the other approaches.
4. All classification approaches could identify "Technology evaluation" ARPs better than "Technology identification" and "Features and configuration" ARPs.

If we compare Fig. 7.2 and Fig. 7.3, we notice that all classification approaches are better at separating ARPs from programming posts (Fig. 7.2) than classifying the three types of ARPs (Fig. 7.3).

7.5.1.3 Analysis of Classification Accuracy

To explain the reasons behind the results in Table 7.5, Fig. 7.2 and 7.3, we created a confusion matrix for the best performing classification approaches A1 to A3. Table 7.6 shows a confusion matrix with true positives for each type of posts (columns) and the results of the classification for each of the best performing classification approaches A1 to A3 (rows). From the confusion matrix,

Table 7.7: Comparing F-scores across different classification approaches using the development sample as training data and testing sample for testing the generalizability of the classification approaches

Pre-processing		**Bag-of-Words**	**Single-Ontology Class**	**Multi-Ontology Class**
Classification Algorithms		**VB5G**	**LMT**	**VB5G**
2-Classes	***P***	0.845	0.824	0.804
	R	0.844	0.8	0.804
	F1	0.844	0.797	0.804
Classification Algorithms		**BN5G**	**LMT**	**VB5G**
4-Classes	***P***	0.658	0.653	0.613
	R	0.656	0.666	0.617
	F1	0.646	0.617	0.604

we can determine the types of posts which the classification approaches could not separate from each other. For example, 1,354 in the first row means that 1,354 posts were correctly classified as programming posts. 53 in the next row means that 53 programming posts were incorrectly classified as "Technology identification" posts. From Table 7.6, we can observe the following:

- The single-ontology-class approach (using LMT) classifies most "Technology identification" ARPs as programming posts, while the multi-ontology-class approach (using VB5G) classifies most "Technology identification" ARPs as either programming posts or "Technology evaluation" ARPs.
- When classifying "Features and configuration" ARPs, all classification approaches falsely classify "Features and configuration" ARPs as programming posts.

7.5.2 Generalisability of Classification Approaches

The proposed classification approaches in the previous sections were trained and tested based on the development sample in Section 7.2.1. The development sample is a sample of posts from the middleware domain. We therefore tested to what degree the proposed classification approaches would generalize to posts from another technology domains.

7.5.2.1 Study Design

We trained the *best performing classification approaches* (A1 to A3 - see Section 7.5.1) on the whole development sample (middleware technologies - see Section 7.2.1) and tested them on the testing sample (big data technologies - see Section 7.2.2). Our goal here is to evaluate the ability of the proposed classification approaches to classify posts from different technology domains. Similar to Section 7.5.1, we calculated the precision P, recall R, and F-score $F1$ for each approach.

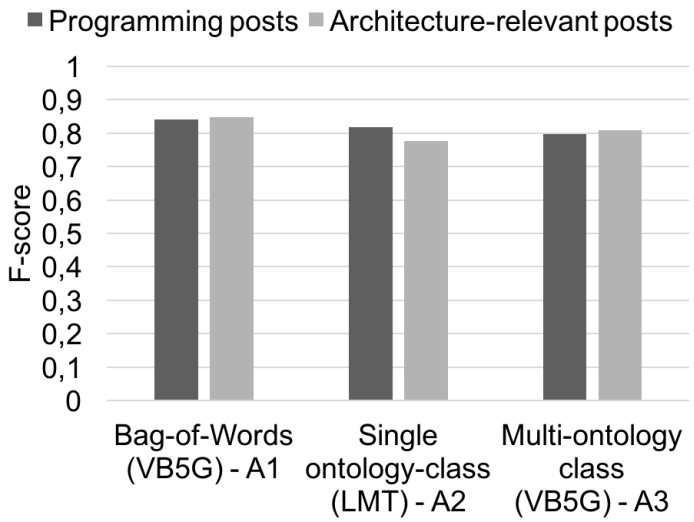


Figure 7.4: Comparing F-scores across different classification approaches and types of posts in the 2-class dataset using the development sample as training data and testing sample for testing the generalizability of the classification approaches

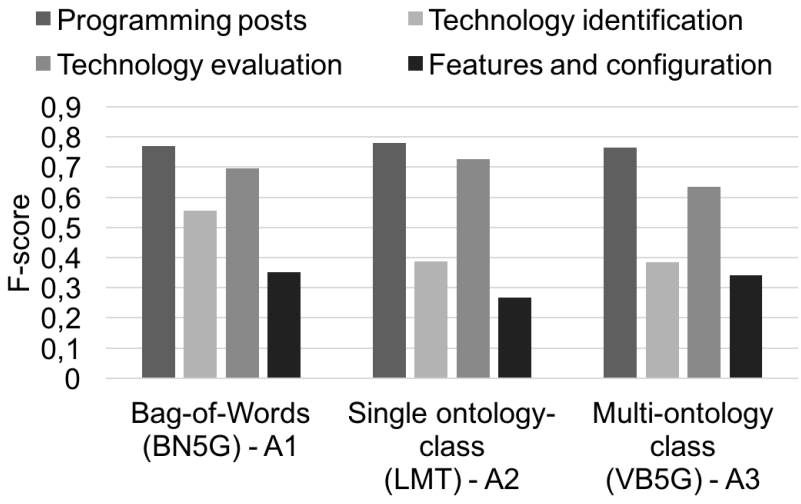


Figure 7.5: Comparing F-scores across different classification approaches and types of posts in the 4-class dataset using the development sample as training data and testing sample for testing the generalizability of the classification approaches

7.5.2.2 Results

Table 7.7 presents the classification results across the different classification approaches. The results using the 2-class data-sets show that the single-ontology class classification approach produce similar classification performance to the results on the development sample of posts, while the bag-of-words and the multi-ontology-class classification approaches show improvement in its performance when testing it on the big data sample. This shows that the classification approaches are stable to potentially perform well for different domains, when identifying architecture-relevant posts and separating them from programming posts. On the other hand, the results in Table 7.7 shows lower accuracy, when using the 4-classes data-sets (i.e. when classifying ARPs into its sub-types). Moreover, the bag-of-words approach shows better accuracy than the single and multi ontology class approaches. This is different from Table 7.5, where the single-ontology-class performs better.

Similar to Section 7.5.1, we calculated precision, recall and F-score for each type of post separately. Fig 7.4 and 7.5 compare the F-score for each type of posts among the 2-class and 4-class data-sets respectively.

From Fig. 7.4, we can observe that all classification approaches could accurately distinguish between big-data programming posts and ARPs. The results show an improvement to identify ARPs over classifying posts from the middelware domain as presented in Fig. 7.2. Moreover, the results in Fig. 7.4 show no significant differences between the different classification approaches. This is also different from the results in Fig. 7.2 when classifying middelware posts.

When comparing the results of classifying middelware posts in Fig. 7.3 with classifying Bigdata posts in Fig. 7.5, we notice the following observations:

- The accuracy of classifying Bigdata posts has been slightly decreased for all types of posts (except the "Technology Evaluation" ARPs).
- All classification approaches could significantly better identify "Technology Evaluation" ARPs, when classifying bigdata posts.
- There are no significant differences between the different classification approaches.

7.6 Significant Terms and Ontology Classes

In order to answer RQ9 and identify terms and ontology classes, which distinguish ARPs from pure programming posts, we performed a quantitative content analysis using a data mining algorithm. We used in our analysis the classified posts from the development sample (858 ARPs, and 1,653 programming posts) in Section 7.2.1.

7.6.1 Significant Terms to Identify ARPs

To identify important terms, which support identifying ARPs, we followed two steps:

1. *Preprocessing of Posts*: We followed a preprocessing similar to Section 7.4.2. To remove the noise from the posts for the data mining algorithms, we first removed stopwords, which

are common words in the English language (e.g., "the", "is"). We used a standard list of stopwords[1]. We also removed numbers and punctuation symbols. Then, we transformed each post section (title, question, and answers) into a "vector of features", which is a vector of 0s and 1s, where each element of a vector shows if a certain word of the list of all words in all posts exist in this post (value of 1) or not (value of 0). We considered the post title, question and answer separately. Thus, each post is associated with three "vectors of features" which represent the existence of words in each of the post sections. In addition, each vector is marked as being either an ARP or programming post.

2. *Terms Selection and Ranking*: We used the "Information Gain Ratio" algorithm [Qui93] provided by the data mining tool Weka 3 [HFH+09] to extract the words, which distinguish ARPs and programming posts. The "Information Gain Ratio" ranges between 0 and 1 and expresses the generative probability ranking of each word with respect to the class (ARP or programming post). In other words, it measures the ability of a word to split the population of posts correctly into the two main classes. Thus, a word with a higher "Information Gain Ratio" has a better ability to classify the posts between ARPs and programming posts because this word is more unique and common for one particular type of post. The algorithm is based on the C 4.5 decision tree learning algorithm, which is one of the most popular algorithms in data mining [WKRQ+07]. As a result from this step, we got a list of all words mentioned in the posts, and their "Information Gain Ratio".

We present in table 7.8 the top 30 terms and ratios derived from the "Information Gain Ratio" algorithm. The distinctive terms in the post title, question, and answers are represented as separate columns in table 7.8, and the gain ratio is associated with each term. The terms are sorted descendingly according to their gain ratio. Appendix C provides an additional list of terms and a complete list of terms is provided in the companion CD. From table 7.8, we notice that many of the mentioned terms might refer to existing architectural concepts. For example, words such as "scalability", "throughput" and "availability" are amongst the top 20 distinctive words, which are used usually to refer to quality attributes, while words such as "pros/cons", "compared" and "decision" might refer to a decision making situation, where the architect needs to decide between two or more architectural solutions. In addition, we notice that technology names (e.g. "xmpp", "tibco", "amqp") as well as architecture patterns (e.g. "broker", "soa", "messaging") are part of the list of words to distinguish ARPs from programming posts. Note that the terms do not express a strict relationship between the terms and architectural concepts. However, the mentioned observations could give an indicator for the existence of such relationships. For instance, many of these terms show up in ARPs as terms, which represent certain ontology classes (see Chapter 5).

7.6.2 Significant Ontology Classes to Identify ARPs

We are specifically interested in ontology classes, because they represent the semantics of text which could be common among different online developer communities. For example, if we want to develop a new classification approach for a new community, a new ontology-based approach could primarily utilize ontology classes with the highest impact on the classification of posts. Similar to the classification approaches, we are interested in simple ontology classes and composite ontology classes. We explore both in the following sub-sections.

[1] http://astellar.com/2011/12/stopwords-for-sphinx-search/

Table 7.8: Top 30 distinctive terms between the ARPs and PPPs in our sample

Title term and gain ratio		**Question term and gain ratio**		**Answer term and gain ratio**	
soa	0.171	scalability	0.176	throughput	0.177
alternatives	0.169	compared	0.175	scaling	0.168
versus	0.167	pros/cons	0.162	xmpp	0.162
comparison	0.151	subscribers	0.162	cloud	0.159
lightweight	0.151	pros	0.159	scalable	0.155
notification	0.151	cons	0.159	tibco	0.148
choosing	0.151	meet	0.154	broker	0.141
distributed	0.148	real-world	0.151	esb	0.14
apps	0.148	amqp	0.151	enterprise	0.14
real-time	0.144	xmpp	0.148	governance	0.14
backend	0.144	mule	0.148	messaging	0.137
oriented	0.144	decision	0.124	brokers	0.136
pros	0.144	soa	0.116	redis	0.136
cons	0.144	corba	0.116	soa	0.132
ready	0.144	messaging	0.11	lightweight	0.129
middleware	0.144	scalable	0.106	scalability	0.128
share	0.14	balancing	0.106	udp	0.128
experience	0.14	availability	0.106	mule	0.128
dto	0.14	dtos	0.106	activemq	0.127
communicate	0.14	alternatives	0.104	bus	0.126
broker	0.14	advantages	0.102	buffers	0.125
layer	0.14	benefits	0.101	scale	0.124
latency	0.14	middleware	0.101	tier	0.124
messaging	0.137	ec2	0.101	ems	0.124
rendezvous	0.135	broker	0.1	systems	0.122
disadvantages	0.135	ipc	0.094	faster	0.121
delivery	0.135	twisted	0.094	subscribers	0.12
xmpp	0.135	technologies	0.092	nservicebus	0.12
bean	0.135	entities	0.088	mature	0.118
payloads	0.135	considering	0.087	benefits	0.118

Table 7.9: Features (Based on Ontology Classes) with Highest IGR

	2-class dataset		4-class dataset	
#	Features	IGR	Features	IGR
1	numCOM	0.118	distQA	0.135
2	distQA	0.117	numCOM	0.128
3	numDIF	0.108	numDIF	0.125
4	hasSourceCode	0.104	hasSourceCode	0.107
5	numQA	0.092	numSUPP	0.106
6	numSUPP	0.091	numQA	0.103
7	numSPED	0.090	numVAL	0.097
8	numVAL	0.084	numSPED	0.094
9	distTEC	0.070	distTEC	0.080
10	numPAT	0.066	numPAT	0.079
11	numCON	0.064	numCON	0.077
12	numREL	0.062	numREL	0.070

7.6.2.1 Simple Ontology Classes and Lexical Triggers

To understand the influence of ontology classes on the quality of the ontology-based classification, we performed the same preprocessing as in Section 7.4.3.1. We then applied the "Information Gain Ratio" (IGR) [Qui93]. We calculated the information gain for both the 2-class and 4-class datasets in the development sample as described in Section 7.2.1. Table 7.9 shows the ranking of the top individual ontology classes in Stack Overflow posts. Complete list for ranking of ontology classes is available in Appendix C. Ontology classes *QA* and *COM* are the most influential simple ontology classes for classifying posts. In addition, several lexical triggers (especially adjectives) have high impact on the classification. When comparing features that count the distinct number of occurrences to their counterparts that count the total number of occurrences (e.g. *numTEC* versus *distNumTEC*, see Section 7.4.3.1) we notice that the distinctive frequency is more suitable to classify posts for ontology classes *QA* and *TEC*.

7.6.2.2 Sequences of Ontology Classes and Composite Ontology Classes

We checked whether sequences of single-ontology classes used as features matched composite ontology classes. This was to ensure that there is indeed a relationship between the ontology classes and sequences of ontology classes found in posts and used as features. We performed here the same preprocessing as in Section 7.4.3.2. We then applied the "Information Gain Ratio" (IGR) [Qui93]. Table 7.10 shows a ranked list (based on "Information Gain Ratio") for the top 10 sequences of ontology classes for both the 2-class and 4-class datasets in the development sample as described in Section 7.2.1. Appendix C and the companion CD provide a complete list of sequences of ontology classes. The majority of sequences matched composing classes for at least one of the composite classes in Chapter 5. Composite ontology classes *FEAT* and *ASTA* are the most frequently appearing classes among the sequences. Composite ontology classes *DR* and *CAS* could not be directly matched with sequences found in posts because these ontology classes are again composed of other composite ontology classes. Composite ontology classes *CONF* and *CB* did not appear among the top sequences at all.

Table 7.10: Features (Based on Sequences of Ontology Classes) with Highest IGR and Their Mapping to Composite Ontology Classes

	2-class dataset			4-class dataset		
#	Sequences (features)	IGR	Matched composite classes	Sequences (features)	IGR	Matched composite classes
1	TEC - PAT	0.085	FEAT	USE - TEC	0.096	UR, ADD
2	TEC - VAL	0.059	ASTA	TEC - QA	0.089	ASTA
3	TEC - QA	0.058	ASTA	TEC - PAT	0.080	FEAT
4	TEC - SUPP	0.055	FEAT	TEC - VAL	0.067	ASTA
5	SUPP - PAT	0.048	FEAT	COM - PAT	0.055	–
6	USE - TEC	0.047	UR, ADD	TEC - WISH	0.054	–
7	TEC - WISH	0.045	–	TEC - DIF	0.049	ASTA
8	TEC - DIF	0.041	ASTA	TEC - SUPP	0.047	FEAT
9	PAT - QA	0.035	–	TEC - COM	0.043	ADD
10	DIF - QA	0.029	ASTA	PAT - COND	0.042	–

7.7 Discussion

7.7.1 Interpretation of Results

Our results show the feasibility of identifying and classifying architecture-relevant posts in developer communities using Stack Overflow as our sample. Our results show high accuracy when identifying ARPs and separating them from programming posts (highest *F1*: 0.842 in Table 7.5) using a limited number of Stack Overflow posts. On the other hand, it is more challenging to further classify ARPs into their subtypes (highest *F1*: 0.734 in Table 7.5). The complexity to classify text on web pages according to certain architecture concepts has been experienced by Gorton et al. [GXY+17], who achieved a maximum precision of 0.59 when trying to detect technology features in online technology product documentation.

Also, our results show that classification approaches based on an ontology classify certain types of posts better than others. For example, the single-ontology-class approach (using LMT) could identify "Technology evaluation" ARPs better than "Technology identification" and "Features and configuration" ARPs (see Fig. 7.3). One reason for this is the ability of an ontology to capture possible vocabulary for certain ontology classes. For example, it is challenging to gather a comprehensive and accurate vocabulary for all terms related to a technology feature (as represented in ontology class *FT*), because there is a high ambiguity between terms that describe technology features [GXY+17]. On the other hand, creating a comprehensive list of technology and pattern names (as represented in ontology classes *TEC* and *PAT*) is more feasible.

Interestingly, well-known AK concepts (as represented in ontology classes *QA*, *ADD*) have the highest influence on classifying architecture posts. Moreover, the usage of adjectives (e.g. represented in ontology classes *DIF*, *SPED*) are important building blocks for architecture discussions in Stack Overflow, because they build argumentation when discussing design decisions (e.g. using ontology class *ASTA*).

The list of distinctive ARP terms show to align with existing software architecture concerns and solutions (e.g. quality attributes, architectural patterns). This possible relationship provides additional validation for our classification between ARPs and programming posts. In addition, these

terms could be a starting point for an additional analysis to ARPs, through analyzing the context of these terms, and relating it to other ontology classes.

7.7.2 Implications of Results on further Research

The proposed classification approaches (and in particular the ensemble learning approach that combines the results of approaches A1 to A3 from Section 7.5.1.2) could be used directly to facilitate populating AK repositories with required information and knowledge. Another usage possibility is to integrate classification approaches with search engines for online developer communities. In this case, the classification approaches could filter and re-rank the results retrieved from search engines based on the architecture relevance of posts. This would improve the effectiveness of pure keyword-based search engines since the ontology used in our classification also considers the semantics of the text in posts. For example, Gottipati et al. [GLJ11] and Zou et al. [ZYL+15] proposed enhanced search engines to search for programming questions in software forums. The engines are based on classifying programming posts into semantically relevant categories using machine learning. However, these approaches did not investigate online developer communities finding AK. In Chapter 8, we used our presented classification approach in this chapter to propose an enhanced search approach. The classifiers identify relevant ARPs to certain design activities. The enhanced search approach shows a significant improvement in the effectiveness of searching compared to a conventional search approach.

8

Enhanced Search Approach for Architectural Information in Developer Communities

8.1 Research Question and Contributions 133
8.2 Overview of Enhanced Search Approach 134
8.3 Evaluation of Search Approach 137
8.4 Discussion 143

8.1 Research Question and Contributions

This chapter provides answer for **RQ10**:
RQ10: How can we improve the search for architecturally relevant information in online developer communities?

By answering RQ10, we support achieving the third goal of the dissertation "Propose approaches to search for architectural information in developer communities" (see Chapter 1). To answer RQ10, we developed an enhanced search approach to search for architecture-relevant information in Stack Overflow. The search approach relies on the proposed classification approach in Chapter 7. Moreover, we evaluate the effectiveness of the new search with practitioners by comparing it to a conventional keyword-based search. We compare the usefulness of the results of both searches for solving architecture design problems as perceived by practitioners.

Our **contributions** are the following:

- We developed a new search approach to help find architecture-relevant information in online developer communities. The main idea of our search approach is to filter and re-rank developer community posts based on their suitability to support architecture design activities.
- We implemented the proposed search approach in a web-based search engine as a proof-of-concept.
- We performed an experiment with 16 practitioners to show the effectiveness of the proposed search approach for supporting architecture design tasks. The experiment shows that the enhanced search outperforms a conventional keyword-bases search.

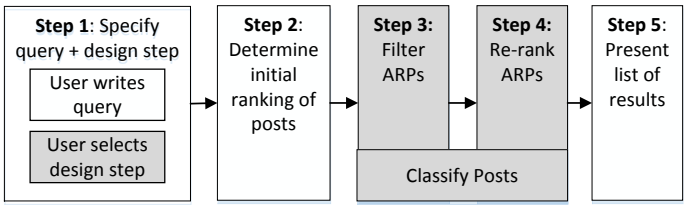


Figure 8.1: Enhanced search process (gray elements are the extensions of a keyword-based search)

Our contributions in this chapter present the first specialized search engine for software architecture. The work provides the next step and extends related work (e.g. Gorton et al. [GXY+17] - see Chapter 2 for further related work) to automatically capture architecture-relevant information from developer communities, textual content of web pages and product documentation. Our work utilizes semantic information captured in our proposed ontology of AK concepts (see Chapter 5), rather than depending solely on lexical features of online developer community posts. In summary, our work shows the feasibility of building specialized search engines for architectural information in online developer communities.

8.2 Overview of Enhanced Search Approach

8.2.1 Steps of Enhanced Search Approach

The most common way to search online developer community pages such as Stack Overflow are web-based search approaches [XBL+17]. Therefore, our search approach also utilizes ideas from keyword-based web search approaches. The basic idea of our approach is to enhance a keyword-based search using two additional steps:

- Filtering and separating "architecture-relevant" posts from other types of posts.
- Re-ranking "architecture-relevant" posts based on their significance to support a certain architecture design step.

These two steps complement a pure keyword-based search with semantic information about architecture relevance of posts. Fig. 8.1 shows the proposed search approach. In detail, our enhanced search consists of five steps (note that Step 3 and Step 4 are our main contributions differentiate our enhanced search from a conventional keyword-based search). We explain each step in the following points:

- **Step 1 – Specify search query + design step:** Similar to a normal keyword-based search, a user defines a query with keywords relevant to a design problem. However, different to a conventional keyword-based search, a user also specifies the current design step. Here, we utilize the three design steps proposed by Kazman and Cervantes [KC16] (see Chapter 2): (1) Identify design concepts, (2) select design concepts, and (3) instantiate architecture elements. These design steps are used to filter and re-rank posts in Step 4 (see below).
- **Step 2 – Determine initial ranking of posts:** This step is the same as in conventional keyword-based search approaches. It first parses and stores data to facilitate fast and accurate information retrieval through mapping between posts and words (note that this indexing

Table 8.1: Frequency of annotations, which belong to relevant ontology classes in different types of posts

Design Step	Relevant Ontology Classes	Technology identification	Technology evaluation	Features and configuration
Identify design concept	ADD for TEC	59	52	30
Select design concepts	ASTA	89	227	32
	DR	12	58	8
	QA	15	14	4
Instantiate architecture elements	ADD for FEAT or CONF	2	3	12

process is not part of our approach; our implementation for this step uses existing libraries, see Section 8.2.3). Then, this step assigns weights to these words based on their ability to differentiate posts (e.g. using the TF/IDF weighting scheme [MRS08]). Finally, based on the keywords in a search query, a similarity score between keywords and the words in posts (considering the weight of each word) is calculated for each post to assess its relevance to the query.

- **Step 3 – Filter architecture-relevant posts:** This step requires information about the type of post, i.e., is a post an architecture-relevant post or not. We obtain this information by classifying posts. Classifying posts as architecture-relevant can be done using various techniques. We use a machine learning approach. Details about the classification are provided in Chapter 7. All architecture-relevant posts with a similarity score higher than zero (as computed in Step 2) are separated from programming posts and moved above programming posts in the list of search results. Architecture-relevant posts are sorted in an ascending order based on their similarity scores.

- **Step 4 – Re-rank architecture-relevant posts:** This step requires information about the type of architecture-relevant post (see types of posts in Chapter 4). We used our proposed classification approach in Chapter 7 to classify the types of ARPs. Filtered ARPs are re-ranked according to their suitability to support the design activity defined in Step 1. To relate types of architecture-relevant posts to design steps, we assigned a "quota" (i.e. estimated percentages) for how types of architecture-relevant posts should appear in the search results depending on the selected design activity. Post types with higher quota have a bigger chance of appearing in the top results. For example, a query for design activity "identify design concepts", the top ten search results will contain 50% of "technology identification" posts, 30% "technology evaluation" posts and 20% of "features and configuration" posts. Posts with the highest similarity scores (from Step 2) are selected as part of the quota for each ARP type. Percentages for "select design concept" and "instantiating architecture elements" are 30%, 50%, 20%, and 10%, 15%, 75%, respectively. These quotas are estimated according to an exploratory study (see Section 8.2.2), and depending on our analysis of ontology classes in Chapter 5.

8.2.2 Relating Types of Architecture-relevant Posts and Design Activities

To determine the most useful types of architecture-relevant posts to support the three generic design steps proposed by Kazman and Cervantes [KC16] (see Chapter 2), we performed an exploratory study in three steps:

1. **Link ontology classes to design steps:** Some ontology classes (see Chapter 5) are more useful than others for certain design steps. Based on the definitions of ontology classes and design steps, we decided what ontology classes are more useful to which design step:
   - For design step “Identifying design concepts”, design decisions on technology solutions are useful to be reused. Therefore, ontology class ADD - "Recommended Design Decision", which contain ontology class TEC - "Technology Solution" has been selected for being relevant to this design step.
   - For design step “Select design concepts”, ontology classes QA - "Quality attribute", ASTA - "Benefits and drawbacks" and DR - "decision rules" are important to evaluate and select solutions [SRZ15].
   - For design step “Instantiate architecture elements”, design decisions on architectural configurations (components and connectors) or technology features are useful to be reused [KC16]. Therefore, ontology class ADD - "Recommended design decision", which contain ontology classes FEAT - "Technology feature" or CONF - "architecture configuration" have been selected for being relevant to this design step.
2. **Calculate occurrences of relevant ontology classes in posts:** In Chapter 5, we annotated 105 architecture-relevant posts, which were classified into the three compact types of ARPs (see Chapter 4). The text in posts of this corpus is annotated with more than 3,800 annotations. Each annotation represents an ontology class as defined in our proposed ontology of (see Chapter 5). Moreover, we counted the number of occurrences of each ontology class for each type of ARP (see Chapter 5). In this step, we are concerned with relevant ontology classes for each design step. Table 8.1 shows the number of occurrences of annotations for each relevant ontology class (categorized by their relevant design step). Calculating the distribution of relevant ontology classes for each type of ARP and design step support estimating useful ARPs for a design step. Chapter 5 provides detailed statistics about the frequency of annotations for all ontology class in the types of posts.
3. **Obtain quotas of post types:** Based on the number of occurrences of ontology classes in the types of posts and the relevance of ontology classes for design activities, we determined quotas for types of posts and how types of posts appear in search results. For example, annotations about ontology classes ASTA and DR (which are important for design step “Select design concepts”) appear in “Technology evaluation” types of posts 1.6 times more frequently than in “Technology identification” posts, and 2.5 times more frequently than in “Features and configuration” posts. Therefore, we defined the quotas 30% “Technology identification” posts, 50% “Technology evaluation” posts and 20% “Features and configuration” posts for design step “Select design concepts”. We estimated the quotas for the other two design steps accordingly. The calculation of the quota support to estimate suitable types of ARPs for each design step.

Table 8.2: Experiences of Practitioners in Experiment

Software development		Software architecture	
# Years	**# Participants**	**# Years**	**# Participants**
>12 Years	6	>5 Years	3
6-12 Years	6	2-5 Years	9
3-5 Years	4	1 Year	4

8.2.3 Implementation of Search Approach

We implemented the enhanced search as a proof-of-concept web-based search engine using Apache Lucene. We used Apache Lucene[1], because it has successfully been applied in previous software engineering research (e.g. Moreno et al. [MBH+15]). The filtering and re-ranking of posts were built on top of the keyword-based search in Lucene. The current version of our implementation includes its own database of posts. This database has been filled by importing posts from Stack Overflow through querying Stack Overflow's API. New posts can easily be added (we provide SQL queries to query Stack Overflow in Appendix A). Similar to conventional search engines, a list of re-ranked posts is presented to the user. The first line of a post and tags assigned to a post are presented in the list. Users would also see the total number of search results and can select the number of results shown on a page, move back and forth between results pages, etc. Clicking on a post leads to the full post. The implementation does not add new usability features compared to what potential users are familiar with from common web-based searches (e.g. Google). Screen-shots are available in Appendix C.

8.3 Evaluation of Search Approach

The primary method to evaluate search approaches is expert judgment about the relevance of the results obtained from a search [MRS08]. Therefore, to compare the effectiveness of the enhanced search approach proposed in this paper (ENHANCED) to a conventional keyword-based search (NORMAL), we conducted an experiment with 16 practitioners (Table 8.2). Here, effectiveness is understood as the ability of a search approach to identify posts relevant for solving architecture design tasks. Thus, we asked participants to perform architecture design tasks, and use two search engines (NORMAL and ENHANCED) to search for information that supports these tasks. To implement NORMAL searches, we used the default implementation of Apache Lucene (i.e., no filtering/re-ranking of search results). We tested the following hypotheses:

- H_0: There is no difference between the effectiveness of NORMAL and ENHANCED.
- H_1: There is a statistically significant difference between the effectiveness of NORMAL and ENHANCED.

[1] http://lucene.apache.org/

8.3.1 Experiment Design

8.3.1.1 Corpus for Experiment

We used our corpus of 2,511 posts from our exploratory study in Chapter 4 that were classified by practitioners. This corpus inlcudes 1,653 programming posts, 282 technology identification posts, 291 technology evaluation posts and 285 features and configuration posts. Furthermore, we classified another 7,702 posts using our classification approach in Chapter 7. We used the best performing classification approach using the ensamble learning to classify posts. These 7,702 posts were randomly selected from a larger pool of posts, which were obtained from Stack Overflow. Like with the 2,511 posts in the original corpus, we chose posts with positive user ratings on Stack Overflow to avoid including poor quality posts in our corpus, because poor quality posts consume much time during a search [XBL+17]. This resulted in a total of 10,213 classified Stack Overflow posts as our corpus for the experiment. This number of posts is similar to other studies in software engineering that investigate search approaches for online communities (e.g. [GLJ11, ZYL+15]).

8.3.1.2 Architecture Design Tasks

Participants in the experiment solved six architecture design tasks, which required them to search for architectural information. Criteria for defining the design tasks were: a) Tasks are independent from the corpus of posts to ensure validity and to prevent bias, and b) tasks should simulate real design problems that occur during different design activities. We based our tasks on the scenarios mentioned by the practitioners during the interview when answering RQ6 and RQ7 (see Chapter 6). The scenarios mentioned by the practioners are previous experiences, which they had to search for architecture information in the past (see Chapter 6 for details on scenarios). This resulted in five tasks. We then categorized these five tasks according to the three design steps from Kazman and Cervantes [KC16] (see Chapter 2). To include two tasks for each design step, we added a sixth task from a case study conducted by Kazman et al. [KC16]. Below we list a brief description for each task for each of the three design activities. A complete description for each task is available in Appendix C.

Tasks for design activity "Identify design concepts":

- T1: For a realtime stock monitoring dashboard, identify suitable middleware technologies which scale to more than 100,000 users.
- T2: A claim management system needs to communicate with mobile apps. Identify JSON parsers for Java with high performance, and taking into consideration organizational policies around open source technologies.

Tasks for design activity "Select design concepts":

- T3: A help desk system communicates with a knowledge base via asynchronous communication and publish/subscribe patterns. Compare interoperability and latency of RabbitMQ, Apache Kafka, and ActiveMQ.
- T4: Compare three technology families for big data systems: Data collector, message brokers, and ETL engines; technologies must support a throughput of 15,000 events/second and ensure availability of 99.99%.

Tasks for design activity "Instantiate architecture elements":

- T5: CRM apps communicate with several other systems using Apache Camel and RabbitMQ. Search for information on technology features and components designs to determine mechanisms for message channeling, translation and routing, as well as a deployment topology (physical design).
- T6: An online shop implemented in Java exposes services to other apps. Search for best practices regarding the right level of service decomposition to achieve high cohesion and low coupling.

To reduce bias or differences between the description of tasks, we captured all tasks in a unified template. The template is guided by templates for developing simulated search tasks for information retrieval systems [Bor03]. Moreover, our proposed conceptual model (see Chapter 6) support determining the main elements of a searching task. Each task includes a description of the functional and quality requirements, constraints, and the actual search goal of a task, which aligns with the design steps.

8.3.1.3 Experiment Execution

We designed the experiment using a Graeco-Latin Square (GLS) design [Sut92]. In this design, the order in which the six design tasks are performed as well as the order in which the two search approaches are used are rotated. Following the GLS design, there are six different sequences of tasks (T1, T2, T3,... T6, then T6, T1, T2,... T5, until sequence T2, T3,... T1). These six sequences are applied twice to rotate the order in which the two search approaches are used (i.e., in total there are 12 sequences of tasks). For the first six sequences (sequences T1, T2,... T6 to T2, T3,... T1), NORMAL was used for the first three tasks in a sequence and ENHANCED for the second three tasks in a sequence. Starting from the seventh sequence to the 12th sequence (i.e., again T1, T2,... T6 to T2, T3,... T1) ENHANCED was used for the first three tasks in a sequence and NORMAL for the second three tasks in a sequence. We randomly assigned participants to sequences of tasks. Since we had 16 participants, every sequence was completed by at least one practitioner, and four sequences were completed by two practitioners.

Each participant read design tasks carefully and then submitted search queries (sequences of words) and obtained a list of ranked posts. We asked participants to analyze only the top ten posts for relevance (users of search engines rarely look beyond the tenth result [GJG04]) and rate each post on a Likert scale from 0 to 3:

1. *Irrelevant (0)*: Post has nothing to do with the task.
2. *Low (1)*: Post contains information which is not immediately relevant to solving the task, but helps refine search.
3. *Medium (2)*: Post addresses a problem different but similar to task at hand, but still provides relevant information.
4. *High (3)*: Post addresses a similar or same problem as specified in the task and contains useful information to solve design task.

The ratings of participants for posts are needed to calculate metrics for search engine effectiveness, which are explained in the next section.

Before executing the experiment with the 16 participants. The study was piloted with four individuals not involved in the actual experiment.

8.3.1.4 Measures of Effectiveness

We measure the effectiveness of each of the search engines using two metrics: *Precision@k* and *Normalized Discount Cumulative Gain*(*nDCG@k*), where k is the maximum number of posts that are considered for evaluation. We considered k from 1 to 10, because the top 10 search results are commonly checked by users of search engines [GJG04].

- *Precision@k* [MRS08] is the ratio between the number of relevant posts (low, medium or high), and the number of retrieved posts in search results m when executing a query, where $m \leq k$. m can be less than k if a poorly phrased search query returns very few search results.
- *nDCG@k*: Precision does not consider the ranking of posts and their degree of relevance (low, medium, high). *nDCG@k* [MRS08] uses weights for posts based on their relevance and ranking in the list of search results (i.e. the higher the relevance [low, medium, high] and rank, the more weight). *nDCG@k* for a query is calculated as

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

with

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

where rel_i is the degree of relevance of a post found by a query (based on the Likert scale above). IDCG@k is the DCG@k value for an ideal ranking of posts for a certain architecture design task. In the above equation for *nDCG@k*, this ideal ranking is based on combining the individual rankings of posts (based on their relevance to a task) from all participants. For example, if user 1 rated posts x, y, z with relevance 3, 3, 1 for task T1 and user 2 rated posts a, b, c with relevance 2, 2, 1 for task T1, then, the ideal ranking for task T1 is 3, 3, 2, 2, 1, 1, 0, 0, 0, 0 when evaluating the top ten search results [MRS08].

8.3.1.5 Exit Survey

After completing all design tasks, we asked participants to fill an exit survey (see Section 8.3.2). We asked participants the following exit questions:

1. How would you evaluate the tasks compared to real architectural design problems?
2. What are the problems you faced when searching for architectural information?
3. Which features would you propose for a tool to improve the search?

The results of the exit survey is presented in Section 8.3.2.

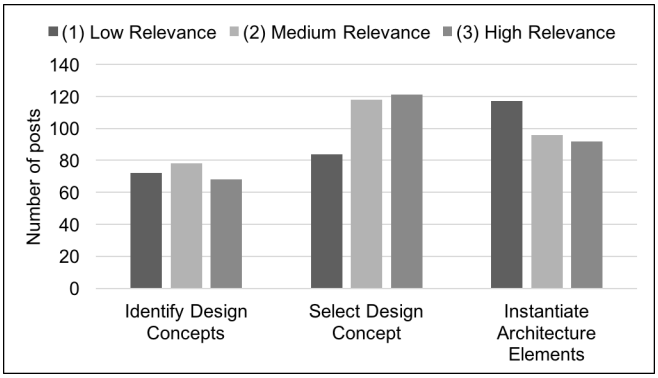


Figure 8.2: Number of relevant posts identified by participants for each design step.

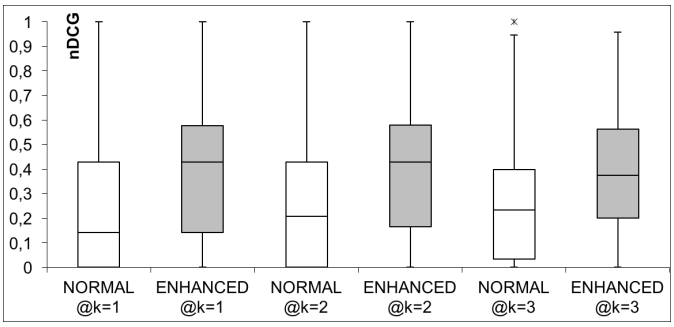


Figure 8.3: nDCG@1-3 for NORMAL and ENHANCED. The lower and upper boxes are the lower and upper quartiles of the distribution, respectively. The vertical thin line is from the minimum to the maximum nDCG values.

8.3.2 Evaluation Results

8.3.2.1 Effectiveness of the Search Approach

Participants submitted a total of 422 different queries (i.e., on average 70 queries per task). The number of queries did not differ significantly between participants, tasks or design activities. The queries contain keywords, which refer to several architecture concepts. For example, the query "Latency and reliability of RabbitMQ" contains keywords which refer to AK concepts like quality attributes and technology solutions. A complete list for the executed queries are available in Appendix C.

Fig. 8.2 shows the number of posts identified by participants relevant to tasks and grouped by their design step. We can observe that queries related to the tasks which belong to the “Select design concept” design step led to the highest number of posts and the highest relevance. On the other hand, queries made for tasks related to the “Identify design concepts” design step yielded the lowest number of posts.

We calculated *Precision*@k and *nDCG*@k for NORMAL and ENHANCED, with @$k_{1\rightarrow10}$. Fig. 8.3 shows a summary for all values of *nDCG* as box-and-whisker plots @$k_{1\rightarrow3}$. The figure shows that ENHANCED has improved the overall nDCG values of searching. Fig. 8.4 and 8.5 show the average *Precision*@k and *nDCG*@k for the three design activities and for ENHANCED and NORMAL. The ENHANCED approach improved the precision of the search for all three design activities. The “Select design concepts” design step achieves the highest *nDCG* improvement, while step “Instantiate architecture elements” has the lowest improvement. Moreover, the values of nDCG for the “Select design concepts” task shows to decrease gradually with the increase

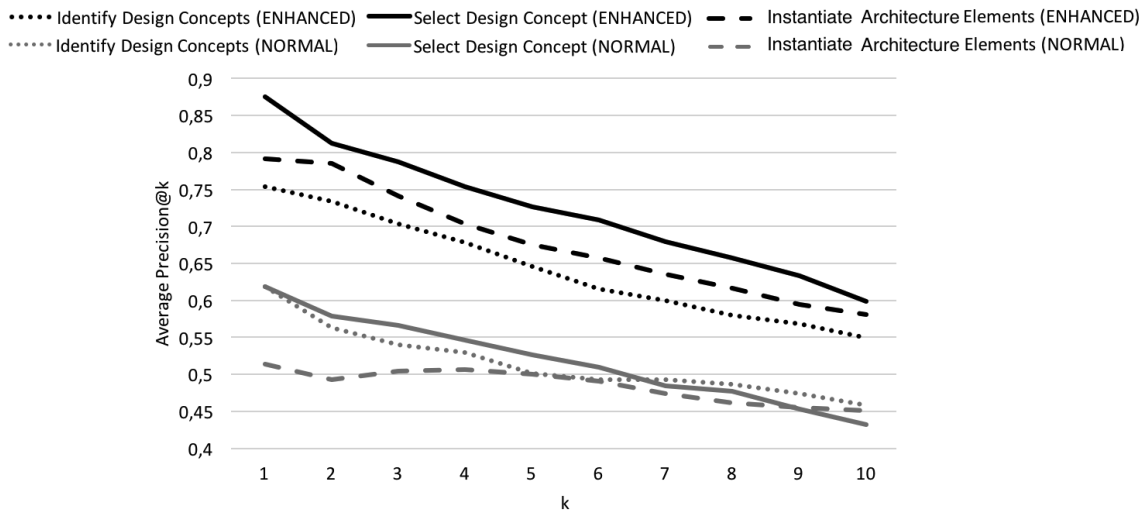


Figure 8.4: Average Precision@k for each design activity

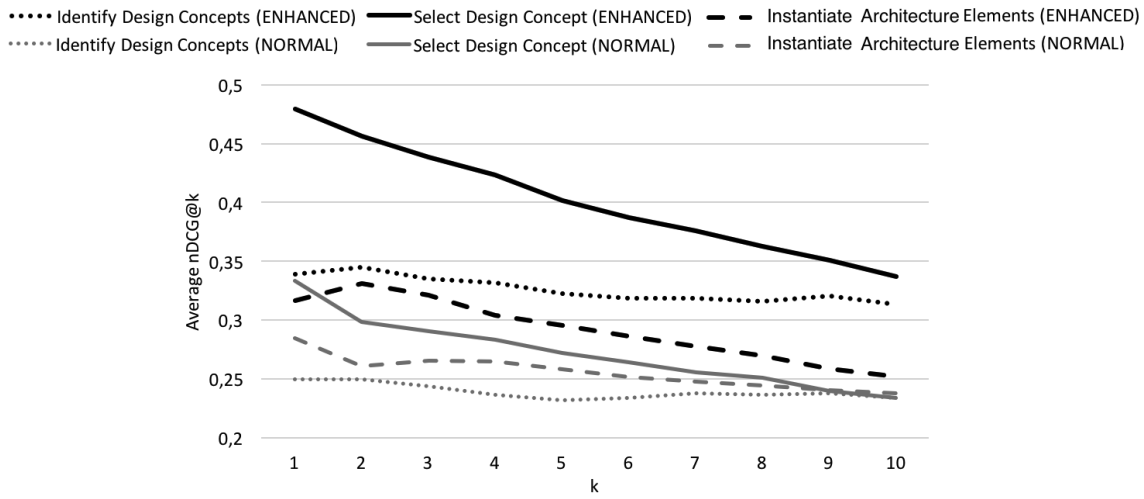


Figure 8.5: Average nDCG@k for each design activity

of k. A possible reason for this could be the ability of the classification approach to identify "Technology evaluation" types of posts better than "Features and configuration" types of posts (see Chapter 7).

To evaluate H_0 and ensure significance, we performed a two-sample T-test with unequal variances [Mon06] on all values of *Precision*@k of ENHANCED and NORMAL and *nDCG*@k for ENHANCED and NORMAL. Fig. 8.6 shows the calculated T-values. The results of the T-test indicate that ENHANCED outperforms NORMAL with statistical significance for both *Precision*@k and *nDCG*@k and for all values of k. From Fig. 8.6, we can observe that the T-values of *Precision* start with high values (highest T-value for *Precision*@$k = 2$) and decrease gradually with the increase of k. On the other hand, the T-values of *nDCG* start with a lower value @$k = 1$ and then increase @$k = 2$, and only change slightly until @$k = 10$.

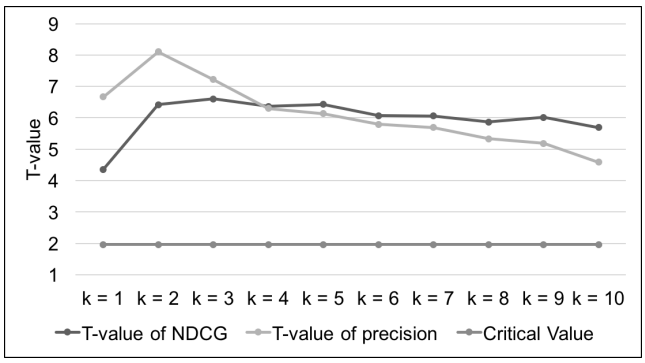


Figure 8.6: Significance T-value for nDCG@k and Precision@k. The Critical T-value is 1.963 (T-values > 1.963 are considered significant). The higher the T-value, the higher the probability for being significant.

8.3.2.2 Results of the Exit Survey

From 16 participants, 12 answered the exit survey. All 12 participants found that design tasks performed in the experiment were realistic and reflect real-life situations. Moreover, 15 from the 16 participants said that they found useful architecture information in posts. The challenges faced by participants were centered around constructing search queries. As one participant wrote, "*Most of the solutions are not in the same level of abstraction as the tasks*". Participants proposed several improvements, such as additional filtering for posts according to certain ontology classes (e.g. TEC, PAT). One participant suggested that "*Similar problems that has the same abstract meaning should be grouped together*". Two of the participants asked for considering the context of problems described in posts to recommend relevant results.

8.4 Discussion

8.4.1 Interpretation of Results

The capability of normal search engines is limited for finding architecture-relevant information in communities. Even with a small search corpus (compared to a search of all posts) and after removing low quality posts, the normal keyword-based search could only reach an average *Precision*@1 (i.e. the first post in the returned list of results is already the most relevant post) between up to 0.51 and 0.62. A real search on millions of posts would yield an even lower precision. This lack of accurate search results from conventional searches might explain the effort required to curate AK [GXY+17]. The lower capability of keywords search engines to find architectural information is a problem, which has been mentioned by practitioners during our interviews in Chapter 6.

Our results show that filtering and re-ranking posts significantly improved the effectiveness of our search to find architecture-relevant information for design problems. In addition, the results show that searching for architectural information for different purposes (i.e. within different design steps) benefit differently from developer communities and the enhanced search. For example, our results in Fig 8.2, 8.4 and 8.5 show that tasks within the "Select design concept" design step find more relevant posts than other design steps. On the other hand, it is challenging to find posts for the "Identify design concepts" tasks. One reason behind this could be the abilities of the classification approach (see Chapter 7). Another possible reason is the complexity to describe a design problem using keywords, because the variations of terms for describing a design problem

is big (e.g. domain and business terms).

8.4.2 Implications of Results on further Research

The proposed search approach provides the first prototype for a specialized search engine to search for software architectural information in developer communities. This motivates proposing new methods towards a comprehensive approach to search for architectural information. For example, a comprehensive searching approach would consider additional sources of knowledge (e.g. Blogs and technology documentations). Our experiment with practitioners to search for architectural information provides initial data (e.g. queries, relevant ARPs), which could be used for further analysis to queries for architectural information and architecture-relevant posts. This could support developing mathematical model to search for architectural information in developer communities. In Chapter 9, we further discuss possible future work based on our results.

Part IV

Conclusion

9

Summary, Discussion and Future Work

9.1 Summary of the Study 146
9.2 Threats to Validity Assessment 148
9.3 Discussion of the Contributions 152
9.4 Future Work 154

In the previous chapters, we explained and justified our research questions, methodologies, and our presented results in details. In this chapter, we provide a summary on the overall research study in the dissertation, which will be presented in Section 9.1. In Section 9.2, we discuss the threats to validity of our results. We discuss then our contributions and their impact on the current state of the art in Section 9.3. Finally, we propose in Section 9.4 several opportunities and needs for future work based on our results.

9.1 Summary of the Study

We started our research with the motivation to support software engineers to take the right design decisions. Taking an architectural design decision is the core of developing a software architecture. Technology design decisions is one of the most important decisions, because they implement several architectural concepts, and they are hard to change after implementation. However, the knowledge about technologies is distributed among several sources of knowledge, such as blogs, forums, and technology documentation. Due to the complexity of acquiring architecture knowledge for technology decisions, we focused in the dissertation on facilitating the acquisition of architecture knowledge for technology design decisions. To address our problem, we conducted several research studies to achieve three main goals, which will be summarized in the following three paragraphs.

First, we had to better understand technology design decisions in practice, because the current state of the art in software architecture did not provide sufficient descriptions and models about technology design decisions. Therefore, we analyzed literature, catalogs of patterns and technology documentation to explore the main building blocks of technologies. We designed our research method to provide maximum support for data gathering and validation from practitioners. Therefore, We interviewed architects from different companies to explore how architects take technology design decision in practice. Our results clarify the main relevant architecture knowledge concepts of a technology design decision and their relationships. Moreover, it integrates different architecture knowledge concepts in literature with technology decisions.

A practically feasible architecture knowledge management solution requires an effective method for knowledge capturing. Current approaches for architecture knowledge management depends on manually documenting and populating architecture knowledge in knowledge repositories, which cannot cope with the fast pace of technology changes. Therefore, we explored developer communities as a practically suitable and effective source for technology-related architecture knowledge. Developer communities provide several advantages over traditional architecture knowledge repositories. For instance, developer communities are continuously up-to-date and extensible with knowledge about new technologies. We selected the biggest and most popular developer community (i.e. Stack Overflow) as our sample to analyze posts in developer communities. Thus, it could well represent knowledge in different developer communities. Our objective was to explore architectural knowledge in Stack Overflow, to structure and classify it in order to mine it as a source for architectural knowledge. We wanted to check if Stack Overflow could be a viable source for reusable architectural knowledge. We selected a sample of posts from Stack Overflow and analyzed them through a systematic qualitative content analysis research process. We first categorized posts in Stack Overflow into several types. We found that there are several categories of architecture-relevant posts in Stack Overflow. We evaluated our classification by comparing it with how practitioners would classify posts and found good agreement. We then selected a sample from the different types of architecture-relevant posts, and annotated their sentences. We associated textual segments with architecture knowledge concepts after several iterations of inter-coder reliability tests. Our results present the first ontology of architecture knowledge in a developer community.

After exploring architecture knowledge in developer communities, we proposed approaches to facilitate capturing architectural knowledge from developer communities. We first interviewed practitioners to understand their searching scenarios and problems to find relevant architectural information in developer communities. Practitioners verified that it is challenging for them to find relevant architectural information in developer communities. Therefore, we proposed classification approaches to automatically identify and classify architecture-relevant posts in developer communities. We systematically compared a set of different classification approaches that do not only rely on keywords, but also consider conceptual and semantic information in posts. The classification approaches were experimented using several methods of pre-processing and classification algorithms. We used our proposed ontology for architecture knowledge to design the features of the classification approaches. This supports capturing the semantics of text when identifying and classifying architecture-relevant posts. The experiments evaluate the accuracy of classification, as well as the generalizability of classification approaches to identify and classify posts in different domains. The results show the feasibility to identify and classify architecture-relevant posts. Based on the classification approaches, we developed, implemented and evaluated an enhanced search approach in developer communities. This search approach does not only rely on keywords, but also considers conceptual and semantic information about architecturally relevant information. The proposed search approach promotes certain types of architecture-relevant posts to their relevant architecture design activity. We evaluated the search approach through an experiment with practitioners, who solved architecture design tasks using both a normal and our proposed enhanced search engines. Comparing our proposed search approach with a conventional search approach showed that the enhanced search leads to more effective results when searching for information during identifying and selecting design concepts, as well as instantiating architecture elements.

9.2 Threats to Validity Assessment

Research results are exposed to threats regarding their validity. Throughout our study, we followed three research methodologies: Interviews, qualitative content analysis, and experiments. Each methodology has different research steps and settings. Therefore, they are exposed to different threats of validity. In the following sub-sections, we discuss the threats to the validity of our results for each research methodology separately. We explain the construct, internal, and external validity threats, as well as reliability of the study.

9.2.1 Threats to Validity for the Interview Studies

We used the interviews research method to determine how practitioners take technology design decisions in practice (answers RQ1 - see Chapter 3). In addition, we used interviews to identify the perspectives of practitioners when using developer communities to search for architectural information (answers RQ6 - see Chapter 6).

Construct Validity:

In this type of validity, we are concerned with validating the accurate representation of the initial content analysis hypothesis through the interviews' questions, as well as the interviews' answers interpretation.

When answering RQ1 in Chapter 3, we mapped our initial hypothesis to the interview questions using several steps: First, we defined a set of general questions, such that each question is related to a hypothesis concept using a concept mapping. After asking the interviewees about their background and experiences, we were able to adapt these questions to align with the experts understanding, which supported a more suitable explication of the hypothesis constructs. The interviewee had no idea about our initial hypothesis during the interview process. In addition, the interview participants work in different companies, and they don't know about each other. This prevented any interactions between the different experts, as well as any possibility of hypothesis guessing.

During the third phase of the interview process when answering RQ1, our main goal was to validate our interpretation for the experiences and examples collected during the second phase of the process. By presenting and explaining our concepts and relating it to the examples mentioned by the participants, we were able to assure that we have the right generalizability across the hypothesis constructs. In addition, conducting the concept validation among all the participants supported us to minimize biasing during the results interpretation.

Internal Validity:

In this type of validity, it is important for us to insure that the interview setup supported us to drive the concluded results. In conducting our interviews as explained Chapters 3 and 6, we followed a set of guidelines (e.g. [HA05]) in questions preparation, as well as in managing the conversation with the participant. With each participant, we started with a general question. For example, when answered RQ1 in Chapter 3, we asked participants: *"IQ: What are the factors which influence choosing a technology?"* during the participant answer, we give the freedom for the expert to explain his answer, and we asked the expert to focus on real experience examples. This supported us to interpret the meaning. In some cases, we mentioned the same question twice, however, with

different ways to ensure that the participant provides the needed information, without interrupting his speaking. When answering RQ1 in Chapter 3, we followed a 3 phase interview process, the 3 phase process helped us to have more than a chance to clarify our understanding to the examples or concepts explained by the interviewees. Even though, it was originally assumed that all experts should have the same level of experience, some interviewed experts contributed to the concepts more than others. For example, when answering RQ1, five of the interviewed experts contributed to the concepts more than the other two. However, this ratio shouldn't impact our results. Moreover, all experts supported us in the model validation through the interview "reflection of meaning" phase.

External Validity:

In this type of validity, we would like to assess our interview study results regarding its generalizability. Regarding this aspect, we have several threats of validity. When answering RQ1 in Chapter 3 and RQ6 in Chapter 6, we did not select the interview participants randomly, as we depend on our network of experts, which might not cover all types of architects in different industries. However, we made no control on their mentioned experiences. As the different participants have experience in different domains, we did not focus our discussion on domain specific problems. For example, when answering RQ1, one of the participants has experience in embedded systems, the interview focused on experiences and technologies used within information and distributed systems domain. Moreover, we focused on the solution level of software architecture during our discussion, more than the enterprise level. Another limitation is the number of conducted interviews, which cannot guarantee *statistical* generalization. However, we aimed for *analytical* generalization. For example, when answering RQ6, the last three interviews confirmed previously identified concepts, but did not lead to new concepts. This gave us confidence that we reached *data saturation* and that we identified the most important concepts. However, we believe that additional empirical studies are needed to extend and validate the proposed concepts.

Reliability:

When answering both RQ1 and RQ6, the author of the dissertation conducted the interviews, and coded the interview discussions. We used then two different methods to ensure reliability of interpretation.

When answering RQ1, we followed an intra-rater reliability method, such that the 3rd phase of the interview ensured that the same concepts have been validated and evaluated by the interviewed participants, while the participants responses were recorded by a single interviewer. All the interviews were conducted one-to-one with prepared questions, which give the chance for each participant to give his opinion about others' inputs. When answering RQ6, a second researcher[1] reviewed and discussed the mapping of concepts and interviewee responses to ensure agreement.

Also, to support replication of the study, as well as further future work, we provide our interview guides in Appendix B.

9.2.2 Threats to Validity for the Content Analysis Studies

We used qualitative content analysis to explore architecture knowledge in the different types of ARPs (answers RQ3 and RQ4 - see Chapter 4), and to determine the textual representation of AK concepts in developer communities (answers RQ5 - see Chapter 5).

[1]The second researcher is Matthias Galster from the university of Canterbury

Internal and Construct Validity:

During our posts collection, we executed several queries, which are based on keywords for middleware technology names. Even though we considered the most popular technologies within the middleware domain, we cannot claim that our list of middleware technologies was exhaustive. There could be other posts related to less popular technologies, which were not included in our list of middleware technologies. However, less popular technologies are most likely not discussed in many posts and therefore should not significantly impact our findings.

External Validity:

External validity is concerned with the generalizability of our analysis results to other posts, which we did not analyze.

One limitation is our conditions for gathering our sample of posts (focusing on middleware posts in a single developer community - Stack Overflow), which might not generalize on all posts in all developer communities. However, our sample community Stack Overflow is the biggest and most popular developer community. Thus, it is the best representative for developer community posts. In addition, our focus on a domain (i.e. middleware) rather than certain technology stack (e.g. Microsoft) make our analysis generalizable among different technology stacks.

Another limitation is the sample size, which might not be representative for all possible ARPs in Stack Overflow. This limitation in the number of posts is caused by the nature of our work, because before posts could be used in our research, we needed to get feedback from practitioners about the architectural relevance of posts as presented in Chapter 4. This manual classification requires experienced practitioners rather than novice or less experienced practitioners. Also, manually classifying posts is very time-consuming: It took practitioners around four hours to classify 100 posts. We could not find other studies about ARPs in Stack Overflow to compare our results with. However, the results show to align with existing software architecture practices (e.g. the solution evaluation and identification post types could align with the process proposed by Hoffmeister et al. [HKN+07]), which might be a good indicator for its generalizability. Moreover, we tried to ensure that the annotated ARPs cover different ARPs through a stratified sampling. All in all, we believe our results provide an initial hypothesis for other future studies on AK in developers community.

During our evaluation phase with practitioners, we did not select our participants randomly, but through our personal contacts. However, we considered several conditions in their selection, such as their architecture and overall technology experience. In addition, evaluation sampling of posts given to practitioners helped us to cover all the type of posts proportional to their size, which made the evaluation more realistic and consequently generalizable.

Reliability:

Reliability is concerned with ensuring that the same measuring steps would produce the same results when repeated. We consider reliability to be paramount for qualitative content analysis, and as a prerequisite for the validity of the results. We considered reliability in each step of our analysis.

We conducted several inter-coder reliability checks when answering both RQ3 in Chapter 4 and RQ5 in Chapter 5. By the end of our categorization, we made a final reliability test, and calculated Cohen's Kappa reliability coefficients [Coh60] among 10% of our sample posts. The reliability coefficient for the classification between architecture and programming is 0.9, while the classifi-

cation of the architecture relevant post types has a reliability coefficient of 0.75. The reliability coefficient for agreement on annotations when developing the ontology is 0.84. This indicates reliability and agreement between classifications beyond chance. In addition to the reliability tests, we described our process in detail, and associated with it the required data for replication. To allow replication of the study, as well as further future work, we provide the following materials in Appendix A and the companion CD: 1) Our Corpus of ARPs. 2) Coding guide for AK in Stack Overflow. 3) The ontology implemented in OWL, as well as keywords, and examples of coded statements.

9.2.3 Threats to Validity for the Experiments

We conducted a set of experiments to evaluate the proposed classification approaches to identify and classify ARPs (answers RQ8 and RQ9 - see Chapter 7). Moreover, we conducted an experiment with practitioners to evaluate the proposed enhanced search approach for architectural information in developer communities (answers RQ10 - see Chapter 8).

Construct Validity:

Construct validity is concerned with the correct execution of the experiments.

Our proposed classification approaches in Chapter 7 rely on using machine learning algorithms, which present a complexity in its application and execution. To ensure proper application of tools to execute the classification algorithms, a researcher[1] in machine learning executed the experiments using a well known and reliable machine learning tool (i.e. Weka).

Internal Validity:

When using ontology-based classification approaches in Chapter 7, we assumed that a word can belong only to one ontology class. However, the same word might belong to multiple ontology classes. Therefore, we reviewed the ontology and found that only a limited number of words (e.g. "layer") could belong to multiple concepts (e.g. "layer" could belong to ontology classes *PAT* and *COM*). Therefore this should not significantly impact our results. The manual classification of posts in the corpus could be another threat to validity. However, the manual categorization of posts in the corpus has been tested for agreement and showed high reliability and agreement beyond chance.

It is challenging to conduct an experiment with human subjects to solve complex problems such as architecture design tasks. In our experiment with practitioners to evaluate the enhanced search in Chapter 8, the experience and background of participants might have influenced the assessment of the architectural relevance of a post. Also, since participants completed the tasks in their own time, we did not have full control on their behavior (e.g. taking a rest or being tired). We tried to mitigate these issues through rotation of the task order. Another threat to validity are the design tasks themselves, which might have influenced participants to use certain keywords. However, we used real design scenarios from practitioners, and independent from the topics of the posts in the corpus of Stack Overflow posts used for the experiment. Moreover, an additional researcher[2] participated in the preparation of the tasks and independent from the author of the dissertation, who analyzed the Stack Overflow posts. This was needed to prevent bias, when preparing the

[1] The researcher in machine learning is Amr Rekaby Salama from the university of Hamburg

[2] The researcher, who participated in the preparation of tasks is Olaf Zimmermann from the University of Applied Sciences, Rapperswil (HSR FHO)

searching tasks.

External Validity:

The development and comparison of classification approaches in Chapter 7 is based on a limited number of posts from a single developer community (Stack Overflow). This risk was partially mitigated by using a 10-fold cross validation for evaluation and comparison. As mentioned previously, the limitation in the number of posts is caused by the nature of our work, because posts in the corpus should be classified by experienced practitioners, which is a tedious and time consuming process.

One limitation when evaluating the search approach in Chapter 8 is the number of participants and tasks to conduct the experiment, and to evaluate the usefulness of the enhanced search approach. However, using a search engine to solve architecture design tasks showed to be time consuming. It took participants between four and eight hours to solve the six tasks. This time is needed by practitioners to understand the task requirements, read Stack Overflow posts carefully and get familiar to the domain. This is also why we hand only two tasks per design activity. Finally, the "quota" assigned to each type of ARP to potentially help with a design step is another threat. If these quotas were changed, then the search might identified other posts as architecture-relevant and made different recommendations to users of our search engine. The lack of statistical studies on searching for architecture information using search engines prevent using mathematical modeling (e.g. [FVC06]) for the re-ranking of posts (rather than using quotas). Conducting a statistical study for mathematical modeling requires the execution and monitoring for thousands of queries using several tasks, which is challenging with the limited time available for architects to conduct experiments. To mitigate this threat, we followed an empirical approach and developed the quota based on an empirical study. Moreover, we fixed quotas for the experiment.

Reliability:

To ensure the repeatability of experiments, we described the experiments in Chapters 7 and 8. We provide the following data in Appendix C, online[1] and in the companion CD. 1) All classified posts in the corpus, 2) The full ontology, 3) the complete list of tags and their analysis to differentiate types of posts. 4) The searching tasks. 5) The repository of posts used in searching. 6) Source code and libraries for classification and searching.

9.3 Discussion of the Contributions

In this section, we discuss our contributions throughout the whole research process in the dissertation. Moreover, we compare our work with the current state of the art, and discuss the differences and the impacts of our results.

Technology design decisions are important and involve several decision factors: Our interviews with practitioners verified the importance of technology design decisions. This align with existing findings in the current state of the art. However, our proposed AK conceptual model additionally clarify architecture knowledge concepts of technology design decisions to show the complexity behind it. We modeled technology design decisions, their features, architectural aspects, and evaluations as extension for existing architecture knowledge models based on empirical evidence from interviews and literature. The proposed approach considered the different design reasoning

[1] https://swk-www.informatik.uni-hamburg.de/~soliman/Dissertation.zip

approaches that the architect use, as well as the different types of technologies and decisions. Selecting a technology involves deciding on its technology features, which implement many of the well known conceptual solutions (e.g. architectural patterns or tactics). At the same time, the availability of certain technology features and the quality of their implementation influence the decision on a certain technology solution. Thus, practitioners focus on the architecture-relevant technology aspects (ASTAs) of technologies (e.g. performance advantage), which shows the benefits and drawbacks of a certain technology compared to other alternative technology solutions.

Developer communities is one source of technology-related architecture knowledge: Our initial content analysis in Chapter 3 shows that there are several sources of technology-related architecture knowledge such as technology documentations and developer communities. The current state of the art on architecture knowledge capture AK in technology documentation. However, our analysis to developer communities in Chapters 4 and 5 shows several advantages of developer communities over technology documentations. For example, developer communities discuss the benefits and drawbacks of technologies compared to each other. Moreover, developer communities are provided and evaluated by practitioners rather than technology vendors. This supports the credibility and validity of knowledge.

Software engineers share their architecture knowledge in developer communities: While developer communities are well know in practice and research for solving programming problems. Our study is the first to discover and verify that software engineers also discuss architectural design issues in developer communities. Our systematic content analysis on Stack Overflow identifies and classifies architecture-relevant posts into several sub-types and variations. The classification together with sub-types provide a structure for capturing architectural knowledge. Moreover, our ontology of architecture knowledge in developer communities specify the representation of architecture knowledge in developer communities.

Software engineers face several challenges to find architectural information using keywords search engines: Our interviews and experiments with practitioners show that software engineers face several obstacles to find architectural information in developer communities. These obstacles emerge from the fact that the amount of architecture-relevant posts are less than other types of posts (e.g. programming posts). Another reason is the limited abilities of generic keywords-based search engines to find relevant architectural-information. Our experiments with practitioners confirmed that searching and curating architecture knowledge from web pages using search engines is complex and time consuming. Our results align with the software architecture state of the art, where researchers experienced problems with search engines, which returned many irrelevant results.

The first corpus of architecture-relevant posts and architecture knowledge ontology from a developer community: Our exploratory study with Stack Overflow produced two useful and first of their kind artifacts. First, we created the first repository of architecture-relevant posts from a developer community. This repository could be used as the fundamental for further research steps to explore other types of developer communities (e.g. Blogs or other forums). Our second artifact is an ontology of architectural knowledge in developer communities. While several researchers proposed ontologies for architecture knowledge, our proposed ontology is the first developed ontology from posts in a developer community. This makes our proposed ontology practically useful, because it operationalize the fuzzy concepts of architecture knowledge into concrete ontology, which specifies a concrete relationship between architecture knowledge concepts and textual segments in Stack Overflow. The benefit of both artifacts (corpus of architecture-relevant posts, and the ontology), that they could be used to develop approaches for tools to capture architecture knowledge from developer communities. For instance, the developed ontology supports implementing core functionality (e.g., information extraction) needed to acquire architecture knowledge from devel-

oper communities.

The first approach to automatically identify and classify architecture-relevant posts in developer communities: Our exploratory study with Stack Overflow showed that developer communities contain a mix of architecture-relevant posts and other types of posts. Moreover, we found that users of developer communities do not mark (i.e. tag) architecture-relevant posts for being architecture-relevant. Therefore, an approach to automatically identify and classify architecture-relevant posts is needed to facilitate finding relevant architectural information in developer communities. We experimented with several classification approaches using several classification algorithms. Moreover, we used ontology-based text classification approaches to capture the semantics of text, which improved the accuracy of classification. Our results show good accuracy when identifying architecture-relevant posts and separate them from programming posts. However, it was challenging for the classification approaches to classify architecture-relevant posts into its sub-types. This is due to the limited number of architecture-relevant posts used for training the algorithms. We additionally determined terms and ontology classes that could differentiate architecture-relevant posts from programming posts. Based on our findings we can support building new tools for automatically capturing architecture knowledge based on mining online communities.

First approach of a specialized search engine for architectural information: Our interviews with practitioners in Chapter 6, as well as recent results in the current state of the art verified the need to develop specialized search approaches for software architecture. We propose the first approach to improve the search for architectural information in developer communities. Our approach makes use of our classification approaches and our architecture knowledge ontology to determine relevant architecture-relevant posts for a certain design activity. Most importantly, the results from experiments with practitioners showed improved effectiveness of searching based on the information needs for different architecture design activities. The improvement is different depending on the design activity.

9.4 Future Work

In this section, we discuss proposals for future work. We base our proposals on our results, challenges, limitations, and experience with practitioners.

Explore additional developer communities for architectural knowledge: In our study, we also selected Stack Overflow as our sample for a developer community. Software engineers might benefit from other types of developer communities. Other developer communities could have different structures. For example, Blogs have a different structure as forums. Thus, other developer communities could have different types of posts and additional architecture knowledge concepts. Moreover, exploring other types and examples of developer communities would further extend and verify our results. Furthermore, exploring additional developer communities would support extending our corpus of architecture-relevant posts and provide additional annotations for ontology classes, which is required to enhance methods for information extraction and information retrieval.

Improve the accuracy of classifying architecture-relevant posts in developer communities: Our evaluation results show that it is challenging to automatically classify architecture-relevant posts. Further techniques and methods are required to improve the accuracy of classification. Using recent advances in text classification (e.g. deep learning) might improve the accuracy of classification. The improvement in the accuracy of classification would support improving searching approaches for architectural information.

Extend and improve the search approach for architectural information: Our proposed search approach in Chapter 8 optimized searching for architecture-relevant posts in Stack Overflow. A search approach with a wider scope could support software architects to find relevant architectural information in several sources of architectural knowledge. For example, considering technology documentation, vendor websites, and architects Blogs would provide a bigger scope of information to software architects. The search approach could be extended to consider these additional sources of architectural knowledge. Moreover, the effectiveness of the search approach could be extended and improved using additional ontology-based approaches, and mathematical modeling methods. Using ontology-based searching and mathematical modeling require bigger number of posts and annotations.

Improve the usability of tools to search for architectural information: Our proposed search approach in Chapter 8 did not include any additional usability aspects compared to a conventional search engine. Our experiment with practitioners in Chapter 8 as well as our interviews in Chapter 6 showed the need to improve the usability of searching to find architectural information. For instance practitioners mentioned that architecture-relevant posts are long and complex to read, which is time consuming to read. Thus, approaches to facilitate reading and capturing architecture knowledge in posts would reduce the time and effort. Another aspect to improve the usability of finding relevant information in posts is to further categorize posts according to architecture knowledge concepts. For example, categorize posts based on their focus on certain quality attributes. Finally, current search approaches require the ability to find suitable keywords. This could be solved if we could provide the ability to search using architecture knowledge concepts (i.e. semantic search) instead of keywords.

Assess the usefulness of a specialized search approach for architectural information: We evaluated the effectiveness of the proposed search approach in Chapter 8. However, we did not assess the impact of our approach on the produced quality of architectures of software engineers. The search approach could be integrated in the tools (e.g. modeling tools) used by software architects. A long-term monitoring for the changes in the quality of software architectures would support evaluating the usefulness of using these specialized search approaches.

Part V

Appendices

A

Stack Overflow Posts Analysis

A.1 Technology Specification Resources 157
A.2 Stack Overflow Posts Repository 158
A.3 Architecture Knowledge Ontology in Stack Overflow 189

A.1 Technology Specification Resources

In Chapter 3, we analyzed several technology specification documents to support answering RQ1 and RQ2. We list here the technology documents, which have been analyzed:

- Sun Certified Enterprise Architect for Java EE Study Guide, Second Edition, Mark Cade and Humphrey Sheil.
- The Java EE 7 Tutorial, Oracle.
- Microsoft Application Architecture Guide, 2nd Edition.
- Microsoft (2016): Windows Communication Foundation. https://docs.microsoft.com/en-us/dotnet/framework/wcf/index
- The Netty Project (2016): Netty Homepage. http://netty.io/.
- The Netty Project 3.x User Guide. http://netty.io/3.5/guide/

```
DECLARE @Text1 VARCHAR(100) = ##Text:string##;
SET     @Text1              = '% '+LTRIM(RTRIM(@Text1))+' %';



SELECT    Id AS [Post Link], Score, Title
FROM       Posts
WHERE
  (PostTypeId = 1) AND
     (Body NOT LIKE '%<pre%><code>%') AND -- for block code with name --
     (Body NOT LIKE '%<pre><code>%') AND -- just block code --
     (Body NOT LIKE '%<pre>%<code>%') AND -- for pre -> new line -> code --
     (Posts.Score >= 7) AND -- change min score -
     ((lower(Posts.Tags) LIKE lower(@Text1)) OR
     (lower(Posts.Title) LIKE lower(@Text1)) OR
     (lower(Posts.Body) LIKE lower(@Text1)))


ORDER BY   Score DESC
```

Figure A.1: The query used to gather stack overflow posts

- Microsoft (2016): Microsoft BizTalk Server 2013 and 2013 R2 Help. https://msdn.microsoft.com/en-us/library/aa548004.aspx
- Hintjens, Pieter: 23/ZMTP - ZeroMQ Message Transport Protocol. http://rfc.zeromq.org/spec:23
- Sustrik, Martin (2014): ZeroMQ: The Design of Messaging Middleware. http://www.drdobbs.com/architecture-and-design/ zeromq-the-design-of-messaging-middlewar /240165684?pgno=1

A.2 Stack Overflow Posts Repository

A.2.1 Queries of Posts

In order to gather Stack Overflow posts to conduct our analysis and experiments in Chapters 4 and 7, we executed queries on the stack exchange web site[1]. The queries retrieve posts, which contain certain technology names in their titles or questions or answers or tags. In Fig. A.1, we provide the query used to gather posts.

We used the query in Fig. A.1 with technology names in the middelware and big-data domains. We list the names of technologies, which are considered to query posts.

For the **middelware** domain, we queried the following technology names:

Zero MQ, wso2, websphere, wcf, wamp, virtuoso, tibco, stomp, starling, soap, rpc, rmi, rabbitmq, ole, netty, mule, msmq, Json WS, jms, gearman, ESB, com, celery, BlazeDS, biztalk, Apache thrift, amqp, Amazon SQS, Activex, .Net Remoting, ActiveMQ, talend, Camel, DDS, hornetq, Json rpc, masstransit, IBM websphere MQ, web orb, Servicemix, Jboss Fuse ESB, ironmq, Apache Avro, kafka, Bpel, mqtt, rpyc, Jboss Messaging MQ, SAP Rfc, kombu, qpid, lightstreamer, Active MQ

For the **big-data** domain, we queried the following technology names:

Mongo DB, Couch DB, Cassandra, NoSQL, Redis, Couchbase, Hadoop, Neo4j

A.2.2 List of Architecture-relevant Stack Overflow Posts

As explained in Chapter 4, we explored different types of architecture-relevant posts. In this section, we present in the following pages a list of the analyzed and categorized architecture relevant posts.

[1] http://data.stackexchange.com/stackoverflow/query/294068/string-search-in-title-tags-body-with-a-minimum-score-of-7-and-without-code-in-bo

URL	Purpose	Solution
http://stackoverflow.com/questions/10006	Identification	Combined
http://stackoverflow.com/questions/10030227	Identification	Configuration
http://stackoverflow.com/questions/10057189	Identification	Feature
http://stackoverflow.com/questions/10156388	Identification	Technology
http://stackoverflow.com/questions/10234386	Identification	Feature
http://stackoverflow.com/questions/1033111	Identification	Feature
http://stackoverflow.com/questions/10355878	Identification	Combined
http://stackoverflow.com/questions/10375137	Identification	Technology
http://stackoverflow.com/questions/1039606	Identification	Technology
http://stackoverflow.com/questions/10568250	Identification	Technology
http://stackoverflow.com/questions/10608531	Identification	Technology
http://stackoverflow.com/questions/10621648	Identification	Technology
http://stackoverflow.com/questions/10745084	Identification	Feature
http://stackoverflow.com/questions/108089	Identification	Combined
http://stackoverflow.com/questions/1103495	Identification	Technology
http://stackoverflow.com/questions/1112279	Identification	Technology
http://stackoverflow.com/questions/1118479	Identification	Configuration
http://stackoverflow.com/questions/1122292	Identification	Technology
http://stackoverflow.com/questions/11290127	Identification	Technology
http://stackoverflow.com/questions/11378046	Identification	Combined
http://stackoverflow.com/questions/1139203	Identification	Configuration
http://stackoverflow.com/questions/11429774	Identification	Combined
http://stackoverflow.com/questions/11481599	Identification	Configuration
http://stackoverflow.com/questions/115316	Identification	Technology
http://stackoverflow.com/questions/1163574	Identification	Technology
http://stackoverflow.com/questions/1164810	Identification	Technology
http://stackoverflow.com/questions/11755146	Identification	Combined
http://stackoverflow.com/questions/11755320	Identification	Feature
http://stackoverflow.com/questions/1183754	Identification	Technology
http://stackoverflow.com/questions/11928655	Identification	Technology
http://stackoverflow.com/questions/11935727	Identification	Technology
http://stackoverflow.com/questions/12147614	Identification	Combined
http://stackoverflow.com/questions/12221715	Identification	Technology
http://stackoverflow.com/questions/1222963	Identification	Feature
http://stackoverflow.com/questions/1234427	Identification	Configuration
http://stackoverflow.com/questions/123817	Identification	Technology
http://stackoverflow.com/questions/12556309	Identification	Combined
http://stackoverflow.com/questions/12585987	Identification	Technology
http://stackoverflow.com/questions/1268252	Identification	Feature
http://stackoverflow.com/questions/12733985	Identification	Technology
http://stackoverflow.com/questions/12783677	Identification	Technology
http://stackoverflow.com/questions/12904377	Identification	Technology
http://stackoverflow.com/questions/13087092	Identification	Technology
http://stackoverflow.com/questions/1314769	Identification	Technology

http://stackoverflow.com/questions/13369521	Identification	Technology
http://stackoverflow.com/questions/13721866	Identification	Feature
http://stackoverflow.com/questions/14186925	Identification	Combined
http://stackoverflow.com/questions/142184	Identification	Technology
http://stackoverflow.com/questions/14342430	Identification	Technology
http://stackoverflow.com/questions/14446384	Identification	Configuration
http://stackoverflow.com/questions/14494444	Identification	Technology
http://stackoverflow.com/questions/14501804	Identification	Technology
http://stackoverflow.com/questions/14663703	Identification	Technology
http://stackoverflow.com/questions/14713711	Identification	Combined
http://stackoverflow.com/questions/1476337	Identification	Feature
http://stackoverflow.com/questions/1479641	Identification	Technology
http://stackoverflow.com/questions/1480462	Identification	Technology
http://stackoverflow.com/questions/1484122	Identification	Technology
http://stackoverflow.com/questions/1485061	Identification	Feature
http://stackoverflow.com/questions/15054777	Identification	Feature
http://stackoverflow.com/questions/15171522	Identification	Technology
http://stackoverflow.com/questions/15265319	Identification	Combined
http://stackoverflow.com/questions/1558207	Identification	Combined
http://stackoverflow.com/questions/1559808	Identification	Technology
http://stackoverflow.com/questions/1570939	Identification	Configuration
http://stackoverflow.com/questions/15995179	Identification	Combined
http://stackoverflow.com/questions/16047344	Identification	Combined
http://stackoverflow.com/questions/1613586	Identification	Feature
http://stackoverflow.com/questions/162376	Identification	Feature
http://stackoverflow.com/questions/16482269	Identification	Feature
http://stackoverflow.com/questions/1670332	Identification	Configuration
http://stackoverflow.com/questions/1700917	Identification	Combined
http://stackoverflow.com/questions/1728020	Identification	Technology
http://stackoverflow.com/questions/1730100	Identification	Technology
http://stackoverflow.com/questions/1746207	Identification	Combined
http://stackoverflow.com/questions/1763099	Identification	Configuration
http://stackoverflow.com/questions/17730905	Identification	Feature
http://stackoverflow.com/questions/17986907	Identification	Combined
http://stackoverflow.com/questions/1809296	Identification	Technology
http://stackoverflow.com/questions/18531072	Identification	Configuration
http://stackoverflow.com/questions/1854323	Identification	Technology
http://stackoverflow.com/questions/187857	Identification	Feature
http://stackoverflow.com/questions/1897050	Identification	Feature
http://stackoverflow.com/questions/1935040	Identification	Feature
http://stackoverflow.com/questions/19758215	Identification	Technology
http://stackoverflow.com/questions/2006624	Identification	Combined
http://stackoverflow.com/questions/2028568	Identification	Technology
http://stackoverflow.com/questions/210446	Identification	Combined
http://stackoverflow.com/questions/2110965	Identification	Technology

http://stackoverflow.com/questions/2114671	Identification	Combined
http://stackoverflow.com/questions/2155759	Identification	Technology
http://stackoverflow.com/questions/2162234	Identification	Technology
http://stackoverflow.com/questions/2172260	Identification	Technology
http://stackoverflow.com/questions/218538	Identification	Technology
http://stackoverflow.com/questions/2198168	Identification	Technology
http://stackoverflow.com/questions/22175482	Identification	Configuration
http://stackoverflow.com/questions/22340	Identification	Combined
http://stackoverflow.com/questions/2244245	Identification	Configuration
http://stackoverflow.com/questions/2244764	Identification	Feature
http://stackoverflow.com/questions/2249715	Identification	Configuration
http://stackoverflow.com/questions/2252085	Identification	Combined
http://stackoverflow.com/questions/2259982	Identification	Feature
http://stackoverflow.com/questions/226953	Identification	Feature
http://stackoverflow.com/questions/22947234	Identification	Feature
http://stackoverflow.com/questions/2300007	Identification	Technology
http://stackoverflow.com/questions/231924	Identification	Technology
http://stackoverflow.com/questions/23431351	Identification	Technology
http://stackoverflow.com/questions/2363157	Identification	Technology
http://stackoverflow.com/questions/236793	Identification	Feature
http://stackoverflow.com/questions/2370573	Identification	Technology
http://stackoverflow.com/questions/2410879	Identification	Technology
http://stackoverflow.com/questions/2429844	Identification	Technology
http://stackoverflow.com/questions/24798569	Identification	Configuration
http://stackoverflow.com/questions/249985	Identification	Combined
http://stackoverflow.com/questions/2500799	Identification	Feature
http://stackoverflow.com/questions/25019324	Identification	Combined
http://stackoverflow.com/questions/2503071	Identification	Configuration
http://stackoverflow.com/questions/2507536	Identification	Technology
http://stackoverflow.com/questions/2518136	Identification	Technology
http://stackoverflow.com/questions/255198	Identification	Technology
http://stackoverflow.com/questions/2554999	Identification	Feature
http://stackoverflow.com/questions/256897	Identification	Technology
http://stackoverflow.com/questions/25875700	Identification	Combined
http://stackoverflow.com/questions/2592609	Identification	Feature
http://stackoverflow.com/questions/2647380	Identification	Combined
http://stackoverflow.com/questions/2675793	Identification	Technology
http://stackoverflow.com/questions/2679024	Identification	Combined
http://stackoverflow.com/questions/2681318	Identification	Feature
http://stackoverflow.com/questions/2697557	Identification	Combined
http://stackoverflow.com/questions/2733565	Identification	Configuration
http://stackoverflow.com/questions/2737685	Identification	Configuration
http://stackoverflow.com/questions/2781872	Identification	Configuration
http://stackoverflow.com/questions/2782076	Identification	Technology
http://stackoverflow.com/questions/2783795	Identification	Technology

http://stackoverflow.com/questions/2804818	Identification	Combined
http://stackoverflow.com/questions/280991	Identification	Feature
http://stackoverflow.com/questions/28298642	Identification	Configuration
http://stackoverflow.com/questions/286614	Identification	Combined
http://stackoverflow.com/questions/2897513	Identification	Configuration
http://stackoverflow.com/questions/2909782	Identification	Technology
http://stackoverflow.com/questions/2936598	Identification	Technology
http://stackoverflow.com/questions/2943935	Identification	Feature
http://stackoverflow.com/questions/2947470	Identification	Technology
http://stackoverflow.com/questions/29584618	Identification	Combined
http://stackoverflow.com/questions/29782279	Identification	Feature
http://stackoverflow.com/questions/3016446	Identification	Combined
http://stackoverflow.com/questions/3017101	Identification	Combined
http://stackoverflow.com/questions/30278054	Identification	Combined
http://stackoverflow.com/questions/3063321	Identification	Technology
http://stackoverflow.com/questions/3115191	Identification	Technology
http://stackoverflow.com/questions/314258	Identification	Feature
http://stackoverflow.com/questions/314268	Identification	Combined
http://stackoverflow.com/questions/3159673	Identification	Technology
http://stackoverflow.com/questions/3172094	Identification	Technology
http://stackoverflow.com/questions/3186783	Identification	Technology
http://stackoverflow.com/questions/3194508	Identification	Feature
http://stackoverflow.com/questions/3198781	Identification	Configuration
http://stackoverflow.com/questions/3213398	Identification	Technology
http://stackoverflow.com/questions/3267081	Identification	Technology
http://stackoverflow.com/questions/329779	Identification	Technology
http://stackoverflow.com/questions/3400139	Identification	Technology
http://stackoverflow.com/questions/340568	Identification	Technology
http://stackoverflow.com/questions/3413424	Identification	Feature
http://stackoverflow.com/questions/345414	Identification	Combined
http://stackoverflow.com/questions/3465675	Identification	Feature
http://stackoverflow.com/questions/3471013	Identification	Technology
http://stackoverflow.com/questions/3486426	Identification	Technology
http://stackoverflow.com/questions/3513112	Identification	Combined
http://stackoverflow.com/questions/3594675	Identification	Feature
http://stackoverflow.com/questions/3624568	Identification	Technology
http://stackoverflow.com/questions/3640292	Identification	Technology
http://stackoverflow.com/questions/3650457	Identification	Technology
http://stackoverflow.com/questions/3650553	Identification	Feature
http://stackoverflow.com/questions/3689161	Identification	Technology
http://stackoverflow.com/questions/3708853	Identification	Configuration
http://stackoverflow.com/questions/37579	Identification	Technology
http://stackoverflow.com/questions/3758576	Identification	Combined
http://stackoverflow.com/questions/3906356	Identification	Feature
http://stackoverflow.com/questions/393580	Identification	Feature

http://stackoverflow.com/questions/3936358	Identification	Feature
http://stackoverflow.com/questions/393727	Identification	Technology
http://stackoverflow.com/questions/393996	Identification	Technology
http://stackoverflow.com/questions/39585	Identification	Configuration
http://stackoverflow.com/questions/4033891	Identification	Technology
http://stackoverflow.com/questions/4041321	Identification	Technology
http://stackoverflow.com/questions/4106062	Identification	Technology
http://stackoverflow.com/questions/41353	Identification	Technology
http://stackoverflow.com/questions/4308343	Identification	Feature
http://stackoverflow.com/questions/4322819	Identification	Configuration
http://stackoverflow.com/questions/4335	Identification	Feature
http://stackoverflow.com/questions/4378778	Identification	Combined
http://stackoverflow.com/questions/44005	Identification	Technology
http://stackoverflow.com/questions/4405992	Identification	Technology
http://stackoverflow.com/questions/440688	Identification	Technology
http://stackoverflow.com/questions/4425696	Identification	Technology
http://stackoverflow.com/questions/4444208	Identification	Feature
http://stackoverflow.com/questions/446417	Identification	Technology
http://stackoverflow.com/questions/4473567	Identification	Technology
http://stackoverflow.com/questions/450488	Identification	Technology
http://stackoverflow.com/questions/45086	Identification	Technology
http://stackoverflow.com/questions/4575381	Identification	Technology
http://stackoverflow.com/questions/4615744	Identification	Configuration
http://stackoverflow.com/questions/4620442	Identification	Feature
http://stackoverflow.com/questions/4621715	Identification	Feature
http://stackoverflow.com/questions/4683171	Identification	Combined
http://stackoverflow.com/questions/4700292	Identification	Feature
http://stackoverflow.com/questions/4736980	Identification	Technology
http://stackoverflow.com/questions/4741713	Identification	Configuration
http://stackoverflow.com/questions/476211	Identification	Feature
http://stackoverflow.com/questions/4804374	Identification	Configuration
http://stackoverflow.com/questions/4812778	Identification	Combined
http://stackoverflow.com/questions/4857464	Identification	Feature
http://stackoverflow.com/questions/491207	Identification	Technology
http://stackoverflow.com/questions/5009801	Identification	Feature
http://stackoverflow.com/questions/5021428	Identification	Technology
http://stackoverflow.com/questions/5031606	Identification	Technology
http://stackoverflow.com/questions/5035466	Identification	Technology
http://stackoverflow.com/questions/5040644	Identification	Configuration
http://stackoverflow.com/questions/5052102	Identification	Technology
http://stackoverflow.com/questions/5052958	Identification	Combined
http://stackoverflow.com/questions/506022	Identification	Technology
http://stackoverflow.com/questions/5100643	Identification	Technology
http://stackoverflow.com/questions/5167122	Identification	Technology
http://stackoverflow.com/questions/517376	Identification	Technology

http://stackoverflow.com/questions/5239204	Identification	Combined
http://stackoverflow.com/questions/5349428	Identification	Feature
http://stackoverflow.com/questions/5364673	Identification	Technology
http://stackoverflow.com/questions/5391435	Identification	Feature
http://stackoverflow.com/questions/5461974	Identification	Technology
http://stackoverflow.com/questions/5467998	Identification	Technology
http://stackoverflow.com/questions/5503954	Identification	Combined
http://stackoverflow.com/questions/556488	Identification	Combined
http://stackoverflow.com/questions/5599379	Identification	Configuration
http://stackoverflow.com/questions/5604985	Identification	Technology
http://stackoverflow.com/questions/5619403	Identification	Feature
http://stackoverflow.com/questions/5687650	Identification	Feature
http://stackoverflow.com/questions/57689	Identification	Technology
http://stackoverflow.com/questions/5805969	Identification	Combined
http://stackoverflow.com/questions/5883545	Identification	Feature
http://stackoverflow.com/questions/5973249	Identification	Feature
http://stackoverflow.com/questions/6033342	Identification	Technology
http://stackoverflow.com/questions/604776	Identification	Technology
http://stackoverflow.com/questions/6065062	Identification	Combined
http://stackoverflow.com/questions/6118773	Identification	Combined
http://stackoverflow.com/questions/6233364	Identification	Technology
http://stackoverflow.com/questions/625146	Identification	Technology
http://stackoverflow.com/questions/626766	Identification	Technology
http://stackoverflow.com/questions/6302341	Identification	Combined
http://stackoverflow.com/questions/647816	Identification	Feature
http://stackoverflow.com/questions/6518401	Identification	Technology
http://stackoverflow.com/questions/6542676	Identification	Technology
http://stackoverflow.com/questions/6551718	Identification	Technology
http://stackoverflow.com/questions/660445	Identification	Feature
http://stackoverflow.com/questions/6694338	Identification	Combined
http://stackoverflow.com/questions/6717874	Identification	Feature
http://stackoverflow.com/questions/686452	Identification	Combined
http://stackoverflow.com/questions/6889265	Identification	Technology
http://stackoverflow.com/questions/691111	Identification	Technology
http://stackoverflow.com/questions/6993746	Identification	Technology
http://stackoverflow.com/questions/6995558	Identification	Combined
http://stackoverflow.com/questions/721708	Identification	Feature
http://stackoverflow.com/questions/7284126	Identification	Technology
http://stackoverflow.com/questions/7332029	Identification	Combined
http://stackoverflow.com/questions/7349551	Identification	Technology
http://stackoverflow.com/questions/7364326	Identification	Technology
http://stackoverflow.com/questions/7403585	Identification	Configuration
http://stackoverflow.com/questions/7465120	Identification	Technology
http://stackoverflow.com/questions/7479180	Identification	Technology
http://stackoverflow.com/questions/7493306	Identification	Combined

http://stackoverflow.com/questions/749522	Identification	Technology
http://stackoverflow.com/questions/7583132	Identification	Feature
http://stackoverflow.com/questions/7629694	Identification	Combined
http://stackoverflow.com/questions/7650697	Identification	Combined
http://stackoverflow.com/questions/7664701	Identification	Configuration
http://stackoverflow.com/questions/7817303	Identification	Technology
http://stackoverflow.com/questions/7827698	Identification	Technology
http://stackoverflow.com/questions/7837189	Identification	Technology
http://stackoverflow.com/questions/7874609	Identification	Technology
http://stackoverflow.com/questions/7882660	Identification	Configuration
http://stackoverflow.com/questions/810212	Identification	Technology
http://stackoverflow.com/questions/8148702	Identification	Configuration
http://stackoverflow.com/questions/8202282	Identification	Combined
http://stackoverflow.com/questions/8257652	Identification	Technology
http://stackoverflow.com/questions/8309780	Identification	Feature
http://stackoverflow.com/questions/83604	Identification	Configuration
http://stackoverflow.com/questions/84269	Identification	Technology
http://stackoverflow.com/questions/843529	Identification	Combined
http://stackoverflow.com/questions/8523650	Identification	Technology
http://stackoverflow.com/questions/861022	Identification	Technology
http://stackoverflow.com/questions/8653146	Identification	Feature
http://stackoverflow.com/questions/8655252	Identification	Feature
http://stackoverflow.com/questions/881754	Identification	Technology
http://stackoverflow.com/questions/8867881	Identification	Feature
http://stackoverflow.com/questions/890938	Identification	Combined
http://stackoverflow.com/questions/9033409	Identification	Feature
http://stackoverflow.com/questions/9074411	Identification	Combined
http://stackoverflow.com/questions/9118367	Identification	Feature
http://stackoverflow.com/questions/9151698	Identification	Feature
http://stackoverflow.com/questions/9157432	Identification	Technology
http://stackoverflow.com/questions/9167663	Identification	Feature
http://stackoverflow.com/questions/922116	Identification	Feature
http://stackoverflow.com/questions/943712	Identification	Configuration
http://stackoverflow.com/questions/945123	Identification	Combined
http://stackoverflow.com/questions/9453561	Identification	Technology
http://stackoverflow.com/questions/9470013	Identification	Configuration
http://stackoverflow.com/questions/9535421	Identification	Technology
http://stackoverflow.com/questions/9542395	Identification	Technology
http://stackoverflow.com/questions/9577012	Identification	Technology
http://stackoverflow.com/questions/960836	Identification	Technology
http://stackoverflow.com/questions/9783818	Identification	Technology
http://stackoverflow.com/questions/9788572	Identification	Technology
http://stackoverflow.com/questions/9846228	Identification	Technology
http://stackoverflow.com/questions/9874234	Identification	Feature
http://stackoverflow.com/questions/9988343	Identification	Feature

http://stackoverflow.com/questions/10037900	Evaluation	Technology
http://stackoverflow.com/questions/1004295	Evaluation	Technology
http://stackoverflow.com/questions/10051261	Evaluation	Technology
http://stackoverflow.com/questions/10055290	Evaluation	Combined
http://stackoverflow.com/questions/10087396	Evaluation	Configuration
http://stackoverflow.com/questions/100993	Evaluation	Technology
http://stackoverflow.com/questions/10136931	Evaluation	Technology
http://stackoverflow.com/questions/10160463	Evaluation	Configuration
http://stackoverflow.com/questions/10172506	Evaluation	Technology
http://stackoverflow.com/questions/10323393	Evaluation	Feature
http://stackoverflow.com/questions/1037985	Evaluation	Feature
http://stackoverflow.com/questions/10407760	Evaluation	Combined
http://stackoverflow.com/questions/10593739	Evaluation	Technology
http://stackoverflow.com/questions/10641999	Evaluation	Configuration
http://stackoverflow.com/questions/10730519	Evaluation	Technology
http://stackoverflow.com/questions/10849920	Evaluation	Feature
http://stackoverflow.com/questions/1098473	Evaluation	Technology
http://stackoverflow.com/questions/1102254	Evaluation	Technology
http://stackoverflow.com/questions/11155095	Evaluation	Technology
http://stackoverflow.com/questions/11157190	Evaluation	Combined
http://stackoverflow.com/questions/11173252	Evaluation	Configuration
http://stackoverflow.com/questions/11198389	Evaluation	Configuration
http://stackoverflow.com/questions/11248510	Evaluation	Feature
http://stackoverflow.com/questions/11398656	Evaluation	Configuration
http://stackoverflow.com/questions/11445400	Evaluation	Feature
http://stackoverflow.com/questions/11565906	Evaluation	Combined
http://stackoverflow.com/questions/11710507	Evaluation	Technology
http://stackoverflow.com/questions/1171767	Evaluation	Technology
http://stackoverflow.com/questions/11798325	Evaluation	Feature
http://stackoverflow.com/questions/11829551	Evaluation	Combined
http://stackoverflow.com/questions/11840630	Evaluation	Combined
http://stackoverflow.com/questions/1189420	Evaluation	Feature
http://stackoverflow.com/questions/11987838	Evaluation	Feature
http://stackoverflow.com/questions/12054412	Evaluation	Technology
http://stackoverflow.com/questions/12069653	Evaluation	Technology
http://stackoverflow.com/questions/120791	Evaluation	Technology
http://stackoverflow.com/questions/12099446	Evaluation	Technology
http://stackoverflow.com/questions/12130481	Evaluation	Technology
http://stackoverflow.com/questions/1225851	Evaluation	Feature
http://stackoverflow.com/questions/12296787	Evaluation	Technology
http://stackoverflow.com/questions/123187	Evaluation	Technology
http://stackoverflow.com/questions/12349690	Evaluation	Feature
http://stackoverflow.com/questions/1237649	Evaluation	Technology
http://stackoverflow.com/questions/12559570	Evaluation	Technology
http://stackoverflow.com/questions/12595911	Evaluation	Feature

http://stackoverflow.com/questions/12634965	Evaluation	Technology
http://stackoverflow.com/questions/12680981	Evaluation	Technology
http://stackoverflow.com/questions/127038	Evaluation	Technology
http://stackoverflow.com/questions/12805377	Evaluation	Technology
http://stackoverflow.com/questions/12955337	Evaluation	Technology
http://stackoverflow.com/questions/12961471	Evaluation	Technology
http://stackoverflow.com/questions/12982	Evaluation	Configuration
http://stackoverflow.com/questions/12996234	Evaluation	Technology
http://stackoverflow.com/questions/13005410	Evaluation	Technology
http://stackoverflow.com/questions/13016406	Evaluation	Technology
http://stackoverflow.com/questions/13309446	Evaluation	Feature
http://stackoverflow.com/questions/13377992	Evaluation	Technology
http://stackoverflow.com/questions/13440875	Evaluation	Technology
http://stackoverflow.com/questions/13494033	Evaluation	Technology
http://stackoverflow.com/questions/13544792	Evaluation	Technology
http://stackoverflow.com/questions/1365357	Evaluation	Combined
http://stackoverflow.com/questions/13671473	Evaluation	Technology
http://stackoverflow.com/questions/13681213	Evaluation	Technology
http://stackoverflow.com/questions/136856	Evaluation	Technology
http://stackoverflow.com/questions/1369749	Evaluation	Technology
http://stackoverflow.com/questions/13818933	Evaluation	Configuration
http://stackoverflow.com/questions/14050116	Evaluation	Combined
http://stackoverflow.com/questions/1410328	Evaluation	Technology
http://stackoverflow.com/questions/1423065	Evaluation	Technology
http://stackoverflow.com/questions/14241947	Evaluation	Technology
http://stackoverflow.com/questions/1426249	Evaluation	Technology
http://stackoverflow.com/questions/1429318	Evaluation	Technology
http://stackoverflow.com/questions/14491000	Evaluation	Technology
http://stackoverflow.com/questions/1468573	Evaluation	Configuration
http://stackoverflow.com/questions/14773771	Evaluation	Technology
http://stackoverflow.com/questions/1500744	Evaluation	Technology
http://stackoverflow.com/questions/1502298	Evaluation	Technology
http://stackoverflow.com/questions/15056878	Evaluation	Technology
http://stackoverflow.com/questions/1511514	Evaluation	Technology
http://stackoverflow.com/questions/15150133	Evaluation	Technology
http://stackoverflow.com/questions/1518263	Evaluation	Technology
http://stackoverflow.com/questions/15267435	Evaluation	Technology
http://stackoverflow.com/questions/15402233	Evaluation	Technology
http://stackoverflow.com/questions/15480632	Evaluation	Technology
http://stackoverflow.com/questions/15490658	Evaluation	Configuration
http://stackoverflow.com/questions/1560619	Evaluation	Feature
http://stackoverflow.com/questions/1563482	Evaluation	Technology
http://stackoverflow.com/questions/15701263	Evaluation	Technology
http://stackoverflow.com/questions/1582952	Evaluation	Technology
http://stackoverflow.com/questions/16040039	Evaluation	Feature

http://stackoverflow.com/questions/1617505	Evaluation	Configuration
http://stackoverflow.com/questions/16232572	Evaluation	Technology
http://stackoverflow.com/questions/1647225	Evaluation	Technology
http://stackoverflow.com/questions/16532449	Evaluation	Technology
http://stackoverflow.com/questions/16539858	Evaluation	Technology
http://stackoverflow.com/questions/16784172	Evaluation	Technology
http://stackoverflow.com/questions/1679064	Evaluation	Feature
http://stackoverflow.com/questions/16838416	Evaluation	Technology
http://stackoverflow.com/questions/17030615	Evaluation	Configuration
http://stackoverflow.com/questions/1708201	Evaluation	Technology
http://stackoverflow.com/questions/1709047	Evaluation	Feature
http://stackoverflow.com/questions/17098377	Evaluation	Technology
http://stackoverflow.com/questions/17126625	Evaluation	Technology
http://stackoverflow.com/questions/17270863	Evaluation	Technology
http://stackoverflow.com/questions/17401679	Evaluation	Feature
http://stackoverflow.com/questions/1756487	Evaluation	Configuration
http://stackoverflow.com/questions/1763276	Evaluation	Technology
http://stackoverflow.com/questions/17708489	Evaluation	Technology
http://stackoverflow.com/questions/17806977	Evaluation	Technology
http://stackoverflow.com/questions/17814436	Evaluation	Configuration
http://stackoverflow.com/questions/1792958	Evaluation	Configuration
http://stackoverflow.com/questions/18187751	Evaluation	Technology
http://stackoverflow.com/questions/1823705	Evaluation	Technology
http://stackoverflow.com/questions/1824397	Evaluation	Feature
http://stackoverflow.com/questions/18353033	Evaluation	Technology
http://stackoverflow.com/questions/1839576	Evaluation	Feature
http://stackoverflow.com/questions/1840684	Evaluation	Technology
http://stackoverflow.com/questions/1847270	Evaluation	Technology
http://stackoverflow.com/questions/1849989	Evaluation	Feature
http://stackoverflow.com/questions/18521196	Evaluation	Technology
http://stackoverflow.com/questions/18566412	Evaluation	Feature
http://stackoverflow.com/questions/18591999	Evaluation	Technology
http://stackoverflow.com/questions/1859278	Evaluation	Technology
http://stackoverflow.com/questions/19090732	Evaluation	Technology
http://stackoverflow.com/questions/19252777	Evaluation	Technology
http://stackoverflow.com/questions/1941471	Evaluation	Technology
http://stackoverflow.com/questions/1941481	Evaluation	Technology
http://stackoverflow.com/questions/1950764	Evaluation	Technology
http://stackoverflow.com/questions/19560479	Evaluation	Feature
http://stackoverflow.com/questions/2006142	Evaluation	Configuration
http://stackoverflow.com/questions/20091826	Evaluation	Configuration
http://stackoverflow.com/questions/20128124	Evaluation	Technology
http://stackoverflow.com/questions/2013793	Evaluation	Technology
http://stackoverflow.com/questions/20215438	Evaluation	Technology
http://stackoverflow.com/questions/2023130	Evaluation	Technology

http://stackoverflow.com/questions/2028524	Evaluation	Technology
http://stackoverflow.com/questions/20310506	Evaluation	Combined
http://stackoverflow.com/questions/20653240	Evaluation	Configuration
http://stackoverflow.com/questions/2066318	Evaluation	Technology
http://stackoverflow.com/questions/20740114	Evaluation	Technology
http://stackoverflow.com/questions/20773875	Evaluation	Combined
http://stackoverflow.com/questions/2096734	Evaluation	Technology
http://stackoverflow.com/questions/2106715	Evaluation	Feature
http://stackoverflow.com/questions/21098502	Evaluation	Combined
http://stackoverflow.com/questions/2111844	Evaluation	Feature
http://stackoverflow.com/questions/2122176	Evaluation	Technology
http://stackoverflow.com/questions/2166590	Evaluation	Technology
http://stackoverflow.com/questions/217111	Evaluation	Technology
http://stackoverflow.com/questions/21808529	Evaluation	Combined
http://stackoverflow.com/questions/22336838	Evaluation	Technology
http://stackoverflow.com/questions/230427	Evaluation	Technology
http://stackoverflow.com/questions/23240813	Evaluation	Technology
http://stackoverflow.com/questions/23280761	Evaluation	Technology
http://stackoverflow.com/questions/2338661	Evaluation	Feature
http://stackoverflow.com/questions/2344022	Evaluation	Feature
http://stackoverflow.com/questions/23564	Evaluation	Configuration
http://stackoverflow.com/questions/2363397	Evaluation	Technology
http://stackoverflow.com/questions/237794	Evaluation	Feature
http://stackoverflow.com/questions/2392211	Evaluation	Technology
http://stackoverflow.com/questions/2392650	Evaluation	Feature
http://stackoverflow.com/questions/2394112	Evaluation	Technology
http://stackoverflow.com/questions/240471	Evaluation	Technology
http://stackoverflow.com/questions/2426974	Evaluation	Technology
http://stackoverflow.com/questions/2460437	Evaluation	Technology
http://stackoverflow.com/questions/2486721	Evaluation	Technology
http://stackoverflow.com/questions/249200	Evaluation	Technology
http://stackoverflow.com/questions/2498796	Evaluation	Configuration
http://stackoverflow.com/questions/250058	Evaluation	Technology
http://stackoverflow.com/questions/2508361	Evaluation	Configuration
http://stackoverflow.com/questions/252115	Evaluation	Technology
http://stackoverflow.com/questions/2523629	Evaluation	Technology
http://stackoverflow.com/questions/2530880	Evaluation	Technology
http://stackoverflow.com/questions/25323	Evaluation	Configuration
http://stackoverflow.com/questions/25325801	Evaluation	Feature
http://stackoverflow.com/questions/255794	Evaluation	Configuration
http://stackoverflow.com/questions/2567254	Evaluation	Combined
http://stackoverflow.com/questions/2576446	Evaluation	Technology
http://stackoverflow.com/questions/2577744	Evaluation	Technology
http://stackoverflow.com/questions/2592809	Evaluation	Technology
http://stackoverflow.com/questions/2609834	Evaluation	Combined

http://stackoverflow.com/questions/2630492	Evaluation	Technology
http://stackoverflow.com/questions/2634784	Evaluation	Technology
http://stackoverflow.com/questions/2656813	Evaluation	Technology
http://stackoverflow.com/questions/2657586	Evaluation	Technology
http://stackoverflow.com/questions/26623673	Evaluation	Technology
http://stackoverflow.com/questions/2669573	Evaluation	Technology
http://stackoverflow.com/questions/267306	Evaluation	Feature
http://stackoverflow.com/questions/2705043	Evaluation	Technology
http://stackoverflow.com/questions/271326	Evaluation	Technology
http://stackoverflow.com/questions/271504	Evaluation	Technology
http://stackoverflow.com/questions/2715274	Evaluation	Feature
http://stackoverflow.com/questions/2725233	Evaluation	Technology
http://stackoverflow.com/questions/2728315	Evaluation	Technology
http://stackoverflow.com/questions/2751752	Evaluation	Technology
http://stackoverflow.com/questions/27666943	Evaluation	Technology
http://stackoverflow.com/questions/2811741	Evaluation	Technology
http://stackoverflow.com/questions/2812676	Evaluation	Feature
http://stackoverflow.com/questions/2868800	Evaluation	Technology
http://stackoverflow.com/questions/28950	Evaluation	Technology
http://stackoverflow.com/questions/2899271	Evaluation	Feature
http://stackoverflow.com/questions/2955071	Evaluation	Technology
http://stackoverflow.com/questions/296650	Evaluation	Technology
http://stackoverflow.com/questions/3015178	Evaluation	Technology
http://stackoverflow.com/questions/30156407	Evaluation	Configuration
http://stackoverflow.com/questions/3016683	Evaluation	Technology
http://stackoverflow.com/questions/3028899	Evaluation	Technology
http://stackoverflow.com/questions/3055713	Evaluation	Technology
http://stackoverflow.com/questions/309374	Evaluation	Technology
http://stackoverflow.com/questions/3102551	Evaluation	Technology
http://stackoverflow.com/questions/3138003	Evaluation	Technology
http://stackoverflow.com/questions/3151966	Evaluation	Technology
http://stackoverflow.com/questions/3155471	Evaluation	Technology
http://stackoverflow.com/questions/3164821	Evaluation	Technology
http://stackoverflow.com/questions/3202521	Evaluation	Technology
http://stackoverflow.com/questions/3216128	Evaluation	Technology
http://stackoverflow.com/questions/3247331	Evaluation	Technology
http://stackoverflow.com/questions/3280576	Evaluation	Technology
http://stackoverflow.com/questions/32851	Evaluation	Technology
http://stackoverflow.com/questions/3291895	Evaluation	Technology
http://stackoverflow.com/questions/3307516	Evaluation	Technology
http://stackoverflow.com/questions/331419	Evaluation	Configuration
http://stackoverflow.com/questions/3325847	Evaluation	Technology
http://stackoverflow.com/questions/334639	Evaluation	Technology
http://stackoverflow.com/questions/3350860	Evaluation	Technology
http://stackoverflow.com/questions/3355082	Evaluation	Technology

http://stackoverflow.com/questions/3394570	Evaluation	Technology
http://stackoverflow.com/questions/3442278	Evaluation	Technology
http://stackoverflow.com/questions/345749	Evaluation	Combined
http://stackoverflow.com/questions/349717	Evaluation	Technology
http://stackoverflow.com/questions/3547587	Evaluation	Technology
http://stackoverflow.com/questions/35490	Evaluation	Technology
http://stackoverflow.com/questions/3577138	Evaluation	Technology
http://stackoverflow.com/questions/3578218	Evaluation	Technology
http://stackoverflow.com/questions/361491	Evaluation	Technology
http://stackoverflow.com/questions/3624275	Evaluation	Technology
http://stackoverflow.com/questions/3706158	Evaluation	Technology
http://stackoverflow.com/questions/3709987	Evaluation	Combined
http://stackoverflow.com/questions/374464	Evaluation	Technology
http://stackoverflow.com/questions/3751297	Evaluation	Technology
http://stackoverflow.com/questions/3766433	Evaluation	Technology
http://stackoverflow.com/questions/377656	Evaluation	Configuration
http://stackoverflow.com/questions/378088	Evaluation	Technology
http://stackoverflow.com/questions/3792519	Evaluation	Technology
http://stackoverflow.com/questions/380052	Evaluation	Technology
http://stackoverflow.com/questions/3814706	Evaluation	Feature
http://stackoverflow.com/questions/3836900	Evaluation	Configuration
http://stackoverflow.com/questions/3885788	Evaluation	Technology
http://stackoverflow.com/questions/3916983	Evaluation	Technology
http://stackoverflow.com/questions/4078056	Evaluation	Technology
http://stackoverflow.com/questions/409338	Evaluation	Combined
http://stackoverflow.com/questions/4119867	Evaluation	Feature
http://stackoverflow.com/questions/4132759	Evaluation	Technology
http://stackoverflow.com/questions/4163066	Evaluation	Technology
http://stackoverflow.com/questions/4270883	Evaluation	Configuration
http://stackoverflow.com/questions/4293385	Evaluation	Technology
http://stackoverflow.com/questions/429606	Evaluation	Technology
http://stackoverflow.com/questions/4311279	Evaluation	Technology
http://stackoverflow.com/questions/4312343	Evaluation	Feature
http://stackoverflow.com/questions/4344822	Evaluation	Technology
http://stackoverflow.com/questions/4374958	Evaluation	Feature
http://stackoverflow.com/questions/43823	Evaluation	Technology
http://stackoverflow.com/questions/446379	Evaluation	Technology
http://stackoverflow.com/questions/447518	Evaluation	Combined
http://stackoverflow.com/questions/4481131	Evaluation	Feature
http://stackoverflow.com/questions/4485650	Evaluation	Feature
http://stackoverflow.com/questions/4499510	Evaluation	Technology
http://stackoverflow.com/questions/4519963	Evaluation	Feature
http://stackoverflow.com/questions/4559883	Evaluation	Technology
http://stackoverflow.com/questions/4627240	Evaluation	Technology
http://stackoverflow.com/questions/4697740	Evaluation	Technology

http://stackoverflow.com/questions/473932	Evaluation	Technology
http://stackoverflow.com/questions/475794	Evaluation	Technology
http://stackoverflow.com/questions/4769973	Evaluation	Technology
http://stackoverflow.com/questions/4824058	Evaluation	Configuration
http://stackoverflow.com/questions/4870814	Evaluation	Technology
http://stackoverflow.com/questions/4906369	Evaluation	Combined
http://stackoverflow.com/questions/4971437	Evaluation	Configuration
http://stackoverflow.com/questions/502780	Evaluation	Technology
http://stackoverflow.com/questions/507391	Evaluation	Technology
http://stackoverflow.com/questions/507862	Evaluation	Technology
http://stackoverflow.com/questions/512807	Evaluation	Technology
http://stackoverflow.com/questions/5132648	Evaluation	Technology
http://stackoverflow.com/questions/5145129	Evaluation	Technology
http://stackoverflow.com/questions/514598	Evaluation	Technology
http://stackoverflow.com/questions/5165858	Evaluation	Feature
http://stackoverflow.com/questions/5296936	Evaluation	Feature
http://stackoverflow.com/questions/5322675	Evaluation	Combined
http://stackoverflow.com/questions/5385407	Evaluation	Technology
http://stackoverflow.com/questions/5407673	Evaluation	Feature
http://stackoverflow.com/questions/5434585	Evaluation	Technology
http://stackoverflow.com/questions/5444996	Evaluation	Technology
http://stackoverflow.com/questions/5456177	Evaluation	Technology
http://stackoverflow.com/questions/5476056	Evaluation	Configuration
http://stackoverflow.com/questions/5576415	Evaluation	Feature
http://stackoverflow.com/questions/5579853	Evaluation	Technology
http://stackoverflow.com/questions/5592572	Evaluation	Feature
http://stackoverflow.com/questions/561915	Evaluation	Feature
http://stackoverflow.com/questions/5693346	Evaluation	Combined
http://stackoverflow.com/questions/5798473	Evaluation	Technology
http://stackoverflow.com/questions/580858	Evaluation	Technology
http://stackoverflow.com/questions/5809802	Evaluation	Technology
http://stackoverflow.com/questions/5834552	Evaluation	Combined
http://stackoverflow.com/questions/584112	Evaluation	Technology
http://stackoverflow.com/questions/5848069	Evaluation	Technology
http://stackoverflow.com/questions/587899	Evaluation	Configuration
http://stackoverflow.com/questions/594983	Evaluation	Feature
http://stackoverflow.com/questions/5952169	Evaluation	Technology
http://stackoverflow.com/questions/5964599	Evaluation	Feature
http://stackoverflow.com/questions/59677	Evaluation	Feature
http://stackoverflow.com/questions/597397	Evaluation	Configuration
http://stackoverflow.com/questions/5975606	Evaluation	Technology
http://stackoverflow.com/questions/6041183	Evaluation	Technology
http://stackoverflow.com/questions/6061813	Evaluation	Feature
http://stackoverflow.com/questions/6104418	Evaluation	Technology
http://stackoverflow.com/questions/611183	Evaluation	Technology

http://stackoverflow.com/questions/6166746	Evaluation	Technology
http://stackoverflow.com/questions/6169658	Evaluation	Combined
http://stackoverflow.com/questions/617859	Evaluation	Technology
http://stackoverflow.com/questions/622190	Evaluation	Technology
http://stackoverflow.com/questions/6357737	Evaluation	Technology
http://stackoverflow.com/questions/6359717	Evaluation	Technology
http://stackoverflow.com/questions/6418488	Evaluation	Combined
http://stackoverflow.com/questions/6574291	Evaluation	Technology
http://stackoverflow.com/questions/6604050	Evaluation	Feature
http://stackoverflow.com/questions/6636213	Evaluation	Technology
http://stackoverflow.com/questions/6664445	Evaluation	Technology
http://stackoverflow.com/questions/6666	Evaluation	Technology
http://stackoverflow.com/questions/676123	Evaluation	Technology
http://stackoverflow.com/questions/684515	Evaluation	Technology
http://stackoverflow.com/questions/6930236	Evaluation	Technology
http://stackoverflow.com/questions/7044157	Evaluation	Technology
http://stackoverflow.com/questions/7090456	Evaluation	Technology
http://stackoverflow.com/questions/712882	Evaluation	Feature
http://stackoverflow.com/questions/7129821	Evaluation	Technology
http://stackoverflow.com/questions/7340334	Evaluation	Feature
http://stackoverflow.com/questions/7382655	Evaluation	Technology
http://stackoverflow.com/questions/7390453	Evaluation	Technology
http://stackoverflow.com/questions/7390561	Evaluation	Technology
http://stackoverflow.com/questions/7410040	Evaluation	Technology
http://stackoverflow.com/questions/741413	Evaluation	Configuration
http://stackoverflow.com/questions/7425808	Evaluation	Technology
http://stackoverflow.com/questions/7521236	Evaluation	Technology
http://stackoverflow.com/questions/773503	Evaluation	Combined
http://stackoverflow.com/questions/7746830	Evaluation	Technology
http://stackoverflow.com/questions/7803185	Evaluation	Feature
http://stackoverflow.com/questions/785652	Evaluation	Technology
http://stackoverflow.com/questions/7875133	Evaluation	Technology
http://stackoverflow.com/questions/789813	Evaluation	Technology
http://stackoverflow.com/questions/8062212	Evaluation	Technology
http://stackoverflow.com/questions/807692	Evaluation	Technology
http://stackoverflow.com/questions/8145060	Evaluation	Feature
http://stackoverflow.com/questions/8191416	Evaluation	Feature
http://stackoverflow.com/questions/8232194	Evaluation	Technology
http://stackoverflow.com/questions/8261654	Evaluation	Combined
http://stackoverflow.com/questions/8270868	Evaluation	Feature
http://stackoverflow.com/questions/8323950	Evaluation	Technology
http://stackoverflow.com/questions/8406914	Evaluation	Technology
http://stackoverflow.com/questions/848806	Evaluation	Technology
http://stackoverflow.com/questions/8510626	Evaluation	Technology
http://stackoverflow.com/questions/855210	Evaluation	Technology

http://stackoverflow.com/questions/8588309	Evaluation	Combined
http://stackoverflow.com/questions/8677162	Evaluation	Technology
http://stackoverflow.com/questions/876137	Evaluation	Feature
http://stackoverflow.com/questions/8765385	Evaluation	Technology
http://stackoverflow.com/questions/879725	Evaluation	Technology
http://stackoverflow.com/questions/891090	Evaluation	Technology
http://stackoverflow.com/questions/8993467	Evaluation	Technology
http://stackoverflow.com/questions/9031116	Evaluation	Configuration
http://stackoverflow.com/questions/9046887	Evaluation	Technology
http://stackoverflow.com/questions/9060402	Evaluation	Technology
http://stackoverflow.com/questions/9077687	Evaluation	Technology
http://stackoverflow.com/questions/9140299	Evaluation	Technology
http://stackoverflow.com/questions/9140716	Evaluation	Technology
http://stackoverflow.com/questions/9322252	Evaluation	Technology
http://stackoverflow.com/questions/936570	Evaluation	Configuration
http://stackoverflow.com/questions/9502548	Evaluation	Technology
http://stackoverflow.com/questions/9503851	Evaluation	Technology
http://stackoverflow.com/questions/9558128	Evaluation	Technology
http://stackoverflow.com/questions/9607544	Evaluation	Technology
http://stackoverflow.com/questions/9615569	Evaluation	Technology
http://stackoverflow.com/questions/9623482	Evaluation	Technology
http://stackoverflow.com/questions/9696009	Evaluation	Technology
http://stackoverflow.com/questions/969964	Evaluation	Configuration
http://stackoverflow.com/questions/9732381	Evaluation	Technology
http://stackoverflow.com/questions/9742380	Evaluation	Feature
http://stackoverflow.com/questions/976924	Evaluation	Technology
http://stackoverflow.com/questions/9800898	Evaluation	Feature
http://stackoverflow.com/questions/990319	Evaluation	Technology
http://stackoverflow.com/questions/99548	Evaluation	Technology
http://stackoverflow.com/questions/10268613	Multi-purpose	Technology
http://stackoverflow.com/questions/10334306	Multi-purpose	Technology
http://stackoverflow.com/questions/10353575	Multi-purpose	Technology
http://stackoverflow.com/questions/107076	Multi-purpose	Feature
http://stackoverflow.com/questions/10777910	Multi-purpose	Technology
http://stackoverflow.com/questions/1084911	Multi-purpose	Combined
http://stackoverflow.com/questions/1143772	Multi-purpose	Technology
http://stackoverflow.com/questions/1156563	Multi-purpose	Technology
http://stackoverflow.com/questions/1262415	Multi-purpose	Technology
http://stackoverflow.com/questions/1296460	Multi-purpose	Configuration
http://stackoverflow.com/questions/13087058	Multi-purpose	Combined
http://stackoverflow.com/questions/13592894	Multi-purpose	Combined
http://stackoverflow.com/questions/13765969	Multi-purpose	Technology
http://stackoverflow.com/questions/14429203	Multi-purpose	Technology
http://stackoverflow.com/questions/1503900	Multi-purpose	Combined
http://stackoverflow.com/questions/15290984	Multi-purpose	Combined

http://stackoverflow.com/questions/17253618	Multi-purpose	Technology
http://stackoverflow.com/questions/1919472	Multi-purpose	Technology
http://stackoverflow.com/questions/2116799	Multi-purpose	Combined
http://stackoverflow.com/questions/2124221	Multi-purpose	Technology
http://stackoverflow.com/questions/2225761	Multi-purpose	Configuration
http://stackoverflow.com/questions/22459356	Multi-purpose	Technology
http://stackoverflow.com/questions/2279417	Multi-purpose	Combined
http://stackoverflow.com/questions/2474163	Multi-purpose	Combined
http://stackoverflow.com/questions/2536522	Multi-purpose	Feature
http://stackoverflow.com/questions/2594815	Multi-purpose	Technology
http://stackoverflow.com/questions/2613348	Multi-purpose	Combined
http://stackoverflow.com/questions/268129	Multi-purpose	Technology
http://stackoverflow.com/questions/2912051	Multi-purpose	Technology
http://stackoverflow.com/questions/3026345	Multi-purpose	Technology
http://stackoverflow.com/questions/3085811	Multi-purpose	Feature
http://stackoverflow.com/questions/3452527	Multi-purpose	Combined
http://stackoverflow.com/questions/3543835	Multi-purpose	Feature
http://stackoverflow.com/questions/3766150	Multi-purpose	Technology
http://stackoverflow.com/questions/3911147	Multi-purpose	Technology
http://stackoverflow.com/questions/4003102	Multi-purpose	Technology
http://stackoverflow.com/questions/401904	Multi-purpose	Technology
http://stackoverflow.com/questions/4318196	Multi-purpose	Technology
http://stackoverflow.com/questions/4362051	Multi-purpose	Technology
http://stackoverflow.com/questions/4373833	Multi-purpose	Technology
http://stackoverflow.com/questions/461319	Multi-purpose	Technology
http://stackoverflow.com/questions/50114	Multi-purpose	Technology
http://stackoverflow.com/questions/5044585	Multi-purpose	Combined
http://stackoverflow.com/questions/5074129	Multi-purpose	Combined
http://stackoverflow.com/questions/514306	Multi-purpose	Configuration
http://stackoverflow.com/questions/5330172	Multi-purpose	Configuration
http://stackoverflow.com/questions/5469230	Multi-purpose	Technology
http://stackoverflow.com/questions/5757886	Multi-purpose	Combined
http://stackoverflow.com/questions/5879902	Multi-purpose	Combined
http://stackoverflow.com/questions/6008107	Multi-purpose	Technology
http://stackoverflow.com/questions/61437	Multi-purpose	Technology
http://stackoverflow.com/questions/6258547	Multi-purpose	Technology
http://stackoverflow.com/questions/6577218	Multi-purpose	Technology
http://stackoverflow.com/questions/6614343	Multi-purpose	Combined
http://stackoverflow.com/questions/6923497	Multi-purpose	Technology
http://stackoverflow.com/questions/6939110	Multi-purpose	Combined
http://stackoverflow.com/questions/698998	Multi-purpose	Configuration
http://stackoverflow.com/questions/722675	Multi-purpose	Technology
http://stackoverflow.com/questions/7506118	Multi-purpose	Technology
http://stackoverflow.com/questions/7739613	Multi-purpose	Technology
http://stackoverflow.com/questions/7921324	Multi-purpose	Combined

http://stackoverflow.com/questions/8269093	Multi-purpose	Technology
http://stackoverflow.com/questions/8639625	Multi-purpose	Combined
http://stackoverflow.com/questions/899674	Multi-purpose	Configuration
http://stackoverflow.com/questions/9296466	Multi-purpose	Technology
http://stackoverflow.com/questions/934716	Multi-purpose	Configuration
http://stackoverflow.com/questions/1047161	CAPPS	
http://stackoverflow.com/questions/10677493	CAPPS	
http://stackoverflow.com/questions/1139285	CAPPS	
http://stackoverflow.com/questions/11735389	CAPPS	
http://stackoverflow.com/questions/12153451	CAPPS	
http://stackoverflow.com/questions/12227059	CAPPS	
http://stackoverflow.com/questions/12829005	CAPPS	
http://stackoverflow.com/questions/1310414	CAPPS	
http://stackoverflow.com/questions/1345239	CAPPS	
http://stackoverflow.com/questions/14059142	CAPPS	
http://stackoverflow.com/questions/15204051	CAPPS	
http://stackoverflow.com/questions/152338	CAPPS	
http://stackoverflow.com/questions/1778578	CAPPS	
http://stackoverflow.com/questions/17928513	CAPPS	
http://stackoverflow.com/questions/18311816	CAPPS	
http://stackoverflow.com/questions/18353898	CAPPS	
http://stackoverflow.com/questions/18392300	CAPPS	
http://stackoverflow.com/questions/1880109	CAPPS	
http://stackoverflow.com/questions/2045867	CAPPS	
http://stackoverflow.com/questions/205546	CAPPS	
http://stackoverflow.com/questions/20634993	CAPPS	
http://stackoverflow.com/questions/2120690	CAPPS	
http://stackoverflow.com/questions/2131207	CAPPS	
http://stackoverflow.com/questions/2198560	CAPPS	
http://stackoverflow.com/questions/221973	CAPPS	
http://stackoverflow.com/questions/22989833	CAPPS	
http://stackoverflow.com/questions/2314995	CAPPS	
http://stackoverflow.com/questions/2336438	CAPPS	
http://stackoverflow.com/questions/2365702	CAPPS	
http://stackoverflow.com/questions/240373	CAPPS	
http://stackoverflow.com/questions/24602768	CAPPS	
http://stackoverflow.com/questions/2506142	CAPPS	
http://stackoverflow.com/questions/2760839	CAPPS	
http://stackoverflow.com/questions/2809485	CAPPS	
http://stackoverflow.com/questions/2835268	CAPPS	
http://stackoverflow.com/questions/2835595	CAPPS	
http://stackoverflow.com/questions/2851898	CAPPS	
http://stackoverflow.com/questions/2992258	CAPPS	
http://stackoverflow.com/questions/299518	CAPPS	
http://stackoverflow.com/questions/30098076	CAPPS	

http://stackoverflow.com/questions/303911	CAPPS
http://stackoverflow.com/questions/3101026	CAPPS
http://stackoverflow.com/questions/311654	CAPPS
http://stackoverflow.com/questions/3305591	CAPPS
http://stackoverflow.com/questions/331600	CAPPS
http://stackoverflow.com/questions/3629402	CAPPS
http://stackoverflow.com/questions/37969	CAPPS
http://stackoverflow.com/questions/3921436	CAPPS
http://stackoverflow.com/questions/41446	CAPPS
http://stackoverflow.com/questions/4287941	CAPPS
http://stackoverflow.com/questions/434617	CAPPS
http://stackoverflow.com/questions/4486658	CAPPS
http://stackoverflow.com/questions/4679235	CAPPS
http://stackoverflow.com/questions/4801545	CAPPS
http://stackoverflow.com/questions/4840536	CAPPS
http://stackoverflow.com/questions/485084	CAPPS
http://stackoverflow.com/questions/5004958	CAPPS
http://stackoverflow.com/questions/511511	CAPPS
http://stackoverflow.com/questions/51256	CAPPS
http://stackoverflow.com/questions/5287636	CAPPS
http://stackoverflow.com/questions/5474372	CAPPS
http://stackoverflow.com/questions/554382	CAPPS
http://stackoverflow.com/questions/600256	CAPPS
http://stackoverflow.com/questions/6037623	CAPPS
http://stackoverflow.com/questions/6123400	CAPPS
http://stackoverflow.com/questions/6134111	CAPPS
http://stackoverflow.com/questions/6135590	CAPPS
http://stackoverflow.com/questions/6212138	CAPPS
http://stackoverflow.com/questions/6369171	CAPPS
http://stackoverflow.com/questions/6500567	CAPPS
http://stackoverflow.com/questions/651273	CAPPS
http://stackoverflow.com/questions/683745	CAPPS
http://stackoverflow.com/questions/7448677	CAPPS
http://stackoverflow.com/questions/7650788	CAPPS
http://stackoverflow.com/questions/8011220	CAPPS
http://stackoverflow.com/questions/8278887	CAPPS
http://stackoverflow.com/questions/8393845	CAPPS
http://stackoverflow.com/questions/8744953	CAPPS
http://stackoverflow.com/questions/880185	CAPPS
http://stackoverflow.com/questions/9025186	CAPPS
http://stackoverflow.com/questions/9124995	CAPPS
http://stackoverflow.com/questions/9213733	CAPPS
http://stackoverflow.com/questions/9286221	CAPPS
http://stackoverflow.com/questions/9292424	CAPPS
http://stackoverflow.com/questions/9378293	CAPPS

http://stackoverflow.com/questions/952470	CAPPS
http://stackoverflow.com/questions/9569851	CAPPS
http://stackoverflow.com/questions/9977179	CAPPS
http://stackoverflow.com/questions/9985971	CAPPS

A.2.3 Additional examples of Architecture-relevant Stack Overflow Posts

In the next pages, we present additional examples for different variants of architectural relevant posts (ARPs) as defined and categorized in Chapter 4.

ARPs Questions' Variations and Examples

We present additional examples for the different Architecture Relevant Post (ARP) types and variations. Each example is linked with its source in Stackoverflow. So, if you want to check the original post in Stackoverflow, click on the example text.

Table 1: Variants of Solution Evaluation Questions

Comparing solutions
Seeking comparison between technology bundles: 1) T: "Which embedded messaging system -> ActiveMQ or HornetQ", Q: "I would appreciate some general pointers and opinions regarding which of the two messaging systems is: easier to manage, has less gotchas or magic stuff one needs to know and avoid, has less overal dependencies, is simple to work with." 2) Q: "Could someone compare and contrast on WCF Rest services vs. ADO.NET Data Services? What is the difference and when to use which?" 3) Q: "I'm currently using ActiveMQ for my messaging needs; aside from a few db failures, it has worked well. However, I'm at the very least considering trying out RabbitMQ….In what ways does RabbitMQ differ from ActiveMQ? What does RabbitMQ do better or worse than ActiveMQ?...." 4) T: "RIA Services versus WCF services: what is a difference". 5) Q: "Our requirement is very simple. Send messages to users subscribed to a topic. We need our messaging system to be able to support millions of topics and maybe millions of subscribers to any given topic in near real time….what are the advantages or disadvantages of using Redis as a MQ over RabbitMQ." 6) T: "Whats the advantage of using celery with rabbitmq over Redis, MongoDB or Django ORM". 7) Q: "What is the advantage of using new WCF Web API over ASP.NET MVC 3 to expose a lightweight JSON Web service layer?...." 8) Q: "Are there any advantages of using NServiceBus over simply using the .net driver for RabbitMQ" 9) T: "Pros/Cons of using BizTalk instead of NServiceBus or MassTransit". 10) T: "Spread vs MPI vs zeromq?", Q: "why would I choose one over the other?" 11) Q: "....when to use a SOAP based service and when to use a RESTful service….." 12) Q: "Is anyone using Karaf instead of Servicemix? If so, how did you come to this decision?..." 13) T: "Why should I use JMS and not RMI+Queue?" 14) T: "amqp vs amqplib - which Node.js amqp client library is better?" *Seeking comparison between technology features*: 1) T: "IIS WCF service hosting vs Windows Service", Q: "We developed a WCF service and we're looking to deploy it…..We are wondering if is it better to host it

in IIS 7 or with a Windows Service."
2) Q: "What is the difference between a dead letter Queue and a back out queue? In WebSphere MQ terms and in terms of Application Servers."
3) Q: "....I would like to know what is the difference and advantages of claim based over role based authentication…."
4) T: "How would I know if I should use Self-Tracking Entities or DTOs/POCOs?"
5) T: "Which is better: PooledConnectionFactory or CachingConnectionFactory?", Q: "We use Spring (3.2.4) with ActiveMQ (5.8.0) in Tomcat (7.0.41) and it is not clear what the best usage…."
6) T: "WCF - what is the fastest binding?"

Seeking comparison between Architectural Configurations:
1) T: "difference between pub-sub and push-pull pattern in zeroMq"
2) T: "JMS Messaging Performance: Lots of Topics/Queues vs. Extensive Filtering (Message Selectors)"
3) T: "How to best validate JSON on the server-side", Q: "I can see at least two ways: ….validate the JSON upfront as it is, before….try to validate it on-the-go while performing business logic….".
4) T: "Should I make more frequent, smaller calls; or less frequent larger calls?"

Context Independent Solution Assessment

Assessing technology bundles:
1) Q: "I would like to find out about BOTH advantages and disadvantages of Windows Communication Foundation from people who have used it or just know it theoretically."
2) Q: "Is there any real difference to the performance when you use Netty and if you don't use it in an application with tens of thousand of connections?"
3) T: "Is ZeroMQ ready for production use?", Q: "....How are the error messages, documentation….How is the performance?"
4) Q: "....how well IBM MQ can handle transfer of large files (up to 100 MB)? Is it stable? It's from mainframe to UNIX server...".
5) T: "How well will WCF scale to a large number of client users?"

Assessing technology features:
1) Q: "Reading the ActiveMQ documentation (we are using the 5.3 release), I find a section about the possibility of using a JDBC persistence adapter with ActiveMQ. What are the benefits? Does it provide any gain in performance or reliability?"
2) Q: "I have always had the availability of IIS so creating a self-hosted WCF service seems like more work than I would want to do. Why would I want to do this?"
3) T: "Scalability of Duplex Polling with Silverlight / IIS". Q: "does anyone have facts / benchmarks?"
4) T: "Is there generally a noticeable performance hit when calling PInvoke on Win32 / COM methods?"

Assessing Architecture Configuration:

1) Q: “If I am generating POCO objects from EntityFramework, and using these to go to/from the WCF server, is there any reason to create client-side Models for the Views & ViewModels to use instead of just using the POCOs directly?....”
2) Q: “....i was think about publishing the data through OData (WCF Data Service) and query that from the controllers …. and publish the service operations through a WCF service ….. What advantages/disadvantages does this architecture poses?”
3) T: “Is [HTML5 + jQuery] (no ASP.NET) + WCF a valid solution for an enterprise level web application?”

Solution Scenarios and Context:

Technology Bundle Scenarios:

1) T: “What are zeromq use cases?”
2) T: “Are you using BizTalk? If so, how are you using it?” A: “I've worked as a consultant for one the largest oil/energy companies in Europe and they basically use BizTalk for all their messaging/integration stuff. Examples are: Invoices (electronic invoices) sent from and to partners in different formats, sync jobs between AD and third party software that maintains it's own username db and integration between support system and external customers via e-mail. So they have a pretty broad adoption of BizTalk and use a cluster of 5 servers.”
3) Q: “I'm looking into the ESB thing with .net like NServiceBus etc , can someone highlight what kind of real world business problems can be solved….”
4) Q: “I would like to learn what are the scenarios/usecases/ where messaging like RabbitMQ can help consumer web applications….”
5) T: “What is ActiveMQ used for?”

Architecture Configuration Scenarios:

T: “When to use SOA (Service Oriented Architecture)”

Context dependent solution assessment:

Assessing technology bundles:

1) Q: “I'm going to be working on a project that involves a number of elements: ASP.NET MVC website, C# console application, iPhone App….I now need to add an API to the site to allow third parties to select, insert and update records….Should I be going along the line of using the Web API? or because my other applications need a web service, should I stick with a WCF Service?”
2) Q: “I want to create WebChat with realtime sync to DB on server….But about syncing data there is few ways to do it: 1. PHP + websockets by Ratchet ….2. node.js + socket.io ….3. node.js + meteor.js (sockJS)....4. Tornado + TornadIO2 (socket.io) + RabbitMQ….Can anybody say which way is better?”
3) Q: “....I have a scenario where i need to create a application that runs 24x7 picks up mail from a mailbox and create few reports…..I thought i could implement it as

a window service.Is WCF service recommended for this Scenario,any advantage of using it…."

4) Q: "Although I've come across Kafka before, I just recently realized Kafka may perhaps be used as (the basis of) an CQRS, eventstore….anything missing from Kafka for it to be a good eventstore? Would it work? Using it production? …."
5) Q: "We're looking at setting up a MSMQ system with ~8000 clients and one queue per client. On average the system needs to handle ~2000 messages daily from each client, where the message size will range from 1K to MSMQ Max size (4MB). Is this at all possible with MSMQ?...."
6) Q: "I have a web app that will create an image from user input. The image creation could take up to a couple seconds…. If I let the server thread, that is handling the request/response also generate the image, that is going to tie up a thread for a couple seconds, and possibly bog down my server, affect performance, kill puppies, etc….Should I use a task queue, such as Celery, so that the server can hand off the image creation, and go back to handling requests/responses?"

Assessing technology features:

1) T: "WCF Bindings - so many! How do I choose one?", Q: "We have an R Server …. that basically takes a script and a csv file, processes some data and returns results as text….I need to write a service on the R server so that .net clients …. can connect to the R server, submit the script and CSV file, and get the results back….Also, is it best to run the service in IIS, or as a separate "command line" type listener service…."
2) Q: "Our analytic server is written in c++. It basically queries underlying storage engine and returns a fairly big structured data via thrift. A typical requests will take about 0.05 to 0.6 seconds to finish depends on the request size. I noticed that there are a few options in terms of which Thrift server we can use in the c++ code, specifically TNonblockingServer, TThreadedServer, and TThreadPoolServer….How shall I decide which one fits my needs the best?"
3) Q: "I'm thinking on migrating my current service layer based on GWT-RPC to something else. It is about 10 service interfaces with 5 methods each, and involving about 20 different domain entities, so you have an idea of the amount of work that would require to change the whole thing, which obviously I would like to minimize. I'm also using Gilead and a Guice based centralized Servlet to handle all the RPC requests….The options that I'm thinking about are: RequestFactory ….A full JSON/REST approach using RestyGWT….I really would like to get suggestions."

Assessing Architecture Configuration:

1) Q: "I'm putting together a simple asp.net web control, that as the result of an ajax form post inserts a record into a MSQL database. It's possible that the page containing this control will recieve many thousands of hits in a small space of time and I'm worried about the performance.... A solution that has occured to me

is to make the web control place a message into an MSMQ queue and have a Windows Service on the server periodically read the queue and do a batch insert. Does that sound like a sensible architecture given that the web and database servers are running on the same machine....” 2) Q: “....Do you think that building a software application with the Service Oriented Architecture approach by using OTP webservers (Mochiweb) as Services is a good idea? Can the additional XML processing layer destroy all the advantages of such approach?....” 3) T: “Should WCF service typically be singleton or not?”
Technology Solutions Interoperability Assessment:
1) T: “ZooKeeper and RabbitMQ/Qpid together - overkill or a good combination?” 2) Q: “We are trying to get WCF and Java talking to each other using SAML tokens issued from an STS. Despite the fact that both sides are compliant with the standards, WS-Security, WS-Trust, WS-Policy, etc., they don't seem to talk to each other….Has anyone ever been able to make this work?” 3) Q: “My asp.net web pages are on IIS web server and it communicates with WCF services….The performance of my wcf services doesnt seem to be that good and I want to improve the same. Also, I need to balance on scalability as my site will be having a very high traffic….Can i use protobuf API...Please suggest”
Solution Definition and Analysis:
Technology Bundles Definition and Analysis: 1) T: “What is WCF in .NET?” 2) T: “What is RPC framework and Apache Thrift?” 3) Q: “I am trying to understand what JMS and how it is connected to AMQP terminology….” 4) T: “Is BizTalk an ESB?” 5) T: “JMS and ESB - how they are related?” *Architecture Configuration Definition and Analysis:* 6) T: “Can someone explain an Enterprise Service Bus to me in non-buzzspeak?” 7) T: what is a data serialization system?” 8) T: “What is service-oriented architecture?”

Table 2: Variants of Solution Identification Questions

Explicit Technology Solutions Searching

1) Q:"I am currently using the binary formatter (Remoting) to serialize and deserialize objects for sending around my LAN. I have recently upgraded from 2.0 to .NET 3.5….What's the fastest serializer for sending objects across my LAN?..."
2) Q: "Does anybody know of a ESB written in Node.JS….I only need the following features for now: Content based routing, AAA, Logging, Monitoring…."
3) Q: "I'm looking for a small and yet efficient enough lightweight JMS broker solution with no or minimum of dependencies…."
4) T: "What's the best python soap stack for consuming Amazon Web Services WSDL?"
5) Q: "I'm working on a multiplayer game on App Engine and it needs a message queue….is there a memcache-based message queue service that can run on App Engine?"
6) Q: "I am working on a project that involves mobile and web clients with Google's AppEngine PAAS. I would like to use RESTFul webservices with my AppEngine app….It would be helpful to hear experienced developers' views on the frameworks available for this set of platforms and merits versus demerits of each."
7) Q: "I would like to start using Cassandra with a node.js deployment, but I can't find a Thrift or Cassandra client for Node.js and/or JavaScript, Is there one?…."
8) Q: "I'd like to use AMQP to join two services one written in C# and other written in python. I'm expecting quite large volume of messages per second. Is there any AMQP Broker that is production ready?...."
9) T: "Is there a FIFO message queuing service offering the high availability of Amazon SQS?"
10) Q: "Are there any RPC framework implemented with: boost + protobuf?"

Implicit Technology Solutions Searching

Technology Features Examples

1) Q: "Is there a way to setup authentication (ala "Basic Authentication") without actually setting up an SSL Certificate?....", A: "Use TransportCredentialOnly security mode….".
2) Q: "We are using MSMQ right now with WCF activation feature, it enables us not to pull queue to read messages. It like push message to application….we are looking at porting from MSMQ to RabbitMQ...Is there anything in RabbitMQ with .net which can do push notification to subscriber like MSMQ?", A: "In AMQP (and RabbitMQ), there are two ways to retrieve messages: basic.get and basic.consume…."
3) Q: "I have an xmlrpc server using Twisted. The server has a huge amount of data stored in-memory. Is it possible to have a secondary, separate xmlrpc server running which can access the object in-memory in the first server?...."
4) Q: "How do I secure a Java SocketChannel, ServerSocketChannel or, perhaps even, a DatagramChannel with TLS? ….", A: "You need to use the SSLEngine….".
5) Q: "Is it possible to send message via RabbitMQ with some delay?....", A: "With the release of RabbitMQ v2.8, scheduled delivery is now available but as an indirect feature….".

6) Q: “I would like to use thrift with a Java server sending data to a browser using websockets. Is this possible?”, A: “....There is json protocol in thrift which is supported by javascript,....Thrift supports only 2 transports raw tcp, and http….”
7) T: “can netty reliably detect channel close/disconnect?”, A: “One way to solve the provlem is using heartbeat messages. Netty supports heartbeat with IdleStateHandler.”
8) Q: “I am considerung to use zeromq as messaging layer between my applications. At least in some cases I want the communication to be secure and I am thinking about SSL. Is there some standard way how to ssl-enable zeromq?....”, A: “the "accepted solution" would be to implement SSL/TLS over the connection manually, and failing that, use AES 128 or 256 with a secure key sharing mechanism”.

Technology Bundle Examples:
1) Q: “Is there a good way to handle multi-node and multi-core concurrency in Java where the main goal is to leverage the most CPU cycles as possible out of the cluster?”, A: “Depends on what you are doing and your budget you might want to look into …. Norbert, GridGain, Terracotta ….”.
2) Q: “What is a good solution for communication via message broker that supports both (C)Python and Java/JMS applications? My particular requirements are: open source solution, Available on Linux-based systems, ...”, A: “ActiveMQ brokers fully support using the Stomp protocol out of the box …. You can then use any of the python libraries …. to connect to Stomp.”
3) Q: “I have a simple server written in C. It's main purpose is to communicate with some business partners over a proprietary protocol.... I have a number of other processes, however, written in other languages (e.g. Python) that must communicate with the server (locally, on the same Linux server). What are the best options for cross-language IPC in this scenario?”, A: “....Perhaps msgpack over a local socket….”
4) Q: “....I would also like to be able to send my objects back and forth client-server using GWT Remote Procedure Calls (RPC), therefore my objects must be able to "detach". However, GWT RPC serialization cannot handle detached JDO/JPA objects and it doesn't appear as though it will in the near future. My question: what is the simplest and most direct solution to this?....”, A: “I 've recently found Objectify, which is designed to be a replacement for JDO….”.

Alternative Technology Solutions Searching:

1) T: “What are the alternatives to ZeroMQ for moving protocol buffer payloads around?”
2) Q: “We are using ActiveMQ 5.2 as our implementation of choice and we picked it a while ago. It performs well enough for our use right now. Since its been a while, I was wondering what other Java Message Service implementations are in use and why?...”
3) Q: “If you want to use a queuing product for durable messaging under Windows, running .NET 2.0 and above, which alternatives to MSMQ exist today?...”
4) Q: “I want to host web services in an existing C/C++ application. What is the best solution? I would like a solution similar to what JAX-WS does for Java. Specifically

revolving around SOAP requests..."

5) Q: “Is there any C++ network library similar to JBoss's Netty? I need an architecture where I can add protocol handlers to a list and process network packets as objects.”
6) T: “Is there an equivalent to ASP.NET Web API in the Rails world?”

Solutions Recommendation for Requirements and Conceptual Design

Seeking Architecture Configuration Examples:

1) T: “What's the best pattern to design an asynchronous RPC application using Python, Pika and AMQP?”, Q: “....What I need in the end: 1) the consumer [A] subscribes his tasks (around 5k each time) to the cluster. 2) the broker dispatches N messages/requests for each consumer, where N is the number of concurrent tasks it can handle. 3) when a single task is finished, the consumer replies to the broker/producer with the result. 4) the producer receives the replies, update the computation status and, in the end, prints some reports….”, A: “....If your callback is IO-bound (doing lots of networking or disk IO) you can use either threads or a greenlet-based solution….Another option would be to implement your consumer as multiple processes using multiprocessing….”
2) Q: “I have an interesting problem to solve. One of my clients has me developing a stock analysis program with close to 50 years of stock data for almost a thousand symbols. I've developed a series of filters that are applied on any given day to see if anything falls out for a trade. We want to run this filter for each day of data we have for each stock. Basically your begin and end date type report. However it takes 6 minutes to filter each week for each symbol. We are figuring about 40 hours or so to run the report on our entire data set….Does anyone have any general ideas or experiences with a web architecture that will support ultra-long asychronous processes?”, A: “As a general suggestion I would recommend a standalone Windows Service….”.
3) T: “How to layout a queue/worker structure to support large tasks for multiple environments?”, Q: “....we have the following setup: We currently use the default Celery setup. (One queue+exchange called "celery".) Each Task on the queue represents a deployment operation. Each task for an environment ends with a synchronisation phase that potentially takes (very) long….”, A: “I would take a look at zeromq it can do messaging and multi-threading in one super fast library….”.

Seeking Technology Bundle Examples:

1) T: “Which network protocol to use for lightweight notification of remote apps?”, Q: “I have this situation.... Client-initiated SOAP 1.1 communication between one server and let's say, tens of thousands of clients. Clients are external, coming in through our firewall, authenticated by certificate, https, etc.. They can be anywhere, and usually have their own firewalls, NAT routers, etc... They're truely external, not just remote corporate offices. They could be in a corporate/campus network, DSL/Cable, even Dialup….”, A: “The two big parties on multi-tier development in Delphi are components4developers …. and RemObjects ….In

your complex environment, multi-cast UDP might not cut it, but from a overhead perspective it is unbeatable….” 2) Q: “I would like to implement an (open source) web application, where the user sends some kind of request via his browser to a Python web application. The request data is used to define and submit some kind of heavy computing job. Computing jobs are outsourced to a "worker backend" (also Python). During job processing, the job goes through different stages over time (from "submitted" over intermediate states to "finished", ideally). What I would like to accomplish is to display the current job state to the user in real time. This means that the worker backend has to communicate job states back to the web application. The web application then has to push information to the user's browser….Which messaging technology should I apply between web app and worker backend?....”, A: “....I would create a table in that database that is specifically for these jobs….zmq is a good solution for this IMO….”. 3) Q: “I'm working on an online PHP application that has a need for delayed PHP event. Basically I need to be able to execute arbitrary PHP code x many seconds (but it could be days) after the initial hit to a URL. I need fairly precise execution of these PHP event, also I want it to be fairly scalable….” A: “....I came up with two solutions to your problem. PHP/Redis solution …. App Engine's TaskQueue….”.
***Technology Independent Architectural Configuration Searching*:**
1) Q: “For message-oriented middleware that does not consistently support priority messages (such as AMQP) what is the best way to implement priority consumption when queues have only FIFO semantics?....”, A: “Given only FIFO support for a given single queue, you will of course have to introduce either multiple queues, an intermediary, or have a more complex consumer. Multiple queues could be handled in a couple of ways…..”. 2) T: “Which layer of the application should contain DTO implementation”.
***Technology Specific Architectural Configuration Searching*:**
1) Q: “From gearman's main page, they mention running with multiple job servers so if a job server dies, the clients can pick up a new job server….What is the best practice to have high-availability for these servers to make sure jobs aren't interrupted in a failure?....”. 2) Q: “I have been experimenting recently with Silverlight, RIA Services, and Entity Framework using .NET 4.0. I'm trying to figure out if that stack makes sense for use in any of my upcoming projects….my questions: What is the best location for business logic (rules, validations, behaviors, authorization) in an application using this stack?....” 3) Q: “I have a JMS client which is producing messages and sending over a JMS queue to its unique consumer. What I want is more than one consumer getting those messages. ....I would like to get advise on these options and cons / pros that you might see....”.

A.3 Architecture Knowledge Ontology in Stack Overflow

A.3.1 Coding guide

In the next pages, we present the coding guide, which we used to annotate sentences in Stack Overflow architecture-relevant posts (ARPs) as explained in Chapter 5.

Coding Book

Composite Ontology Classes

A. **Design Issue**: expressed in ARPs through description for relevant architecture configurations. The described configurations concern either part of a planned design or an existing software system. Design issues could be described at different levels of abstraction. Most ARP questions provide a long description for the design issue in a couple of sentences, while during answers short references for known design issues are commonly used.
B. **Requirements**: Three types of requirements were found: 1) Quality attribute requirements are mentioned explicitly using the standard quality attribute terms. 2) Technology features requirements show the need of user to have certain technology features as part of the proposed solution. 3) Business requirements are expressed using their business domain terms.
C. **Constraints**: We found three types of constraints: 1) Team skills constraints express the level of knowledge of the user to a certain technology solution. 2) Development time constraint is indicated by expressing explicitly the priority of development time to the user. 3) Solution constraint is also expressed explicitly by indicating which technology solutions must be considered in the solutions.
D. **User Request**: exist in ARP question or title in a form of questions. User request complements design issue, requirements and constraints by showing the type of architecture activity (evaluation or synthesis) considered by the user in this post. The request might embody short references for design issues and requirements.
E. **Technology Solutions Features**: We found two main types of technology features: 1) Development features are expressed through certain programming activities (e.g. debugging) or programming features and tools (e.g. inheritance, code generation), 2) Behavioral features are expressed through technology specific component and class names. In addition, behavioral features could be further classified among different quality attributes (e.g. Interoperability features). Moreover, some of the behavioral features are explained through their implemented architectural patterns or their relationship with other technologies.
F. **Technology Solutions ASTAs:** users mention technology solutions' benefits and drawbacks, as part of their argument for recommending or excluding technology solutions. A key aspect which distinguishes ASTAs is the extensive usage of adjectives and adverbs in combination with technology features and quality attributes. The adjectives or adverbs are used to express the advantages or disadvantages of certain technology solutions or features.
G. **Technology Solutions Use-Cases**: These are either success or failure stories for the usage of technology solutions at certain contexts. The stories could be coming from personal experiences of users, or well-known examples for existing systems. The context associated with stories could include domain description, architecture configurations, infrastructure, and constraints.
H. **Design Decisions:** ADDs came in different forms: 1) Recommended ADDs represent the majority of ADDs. They are recommendation from users based on their experience or opinion for certain architectural solutions. 2) Taken ADDs are ADDs, which have been decided by the user who asked the question. Usually after discussions with other users. 3) Planned and existing system ADDs come usually as part of the design issue description. They represent ADDs which have been previously taken or implemented.
I. **Decision Rules**: Conditional recommendation for architectural solutions. They consists of a rule condition and recommendation. The condition might involve other ontology classes such as requirements, constraints, architectural configuration, and existing system description. On the other hand, recommendations involve recommended ADDs for certain technology solution or architecture configuration.

J. **Architecture configuration**: represents part of an architectural model. The ontology class is represented through a sentence, which consists of one or more component names or application types associated with a connector verb or name. (e.g. “Pushing data from the server to the client”, “Rubby app sends a request to Java app”)
K. **Component behavior**: A sentence which describes the behavior of a component. It give an overview about the type of implemented logic and complexity. In addition, sometimes component interface and internal operations are mentioned during the description. (e.g. “service can be viewed as the business layer of the application”, “process will run asynchronously”).
L. **Existing system description**: describe the architecture of an existing system, which a user is dealing with. The description depends on other ontology classes such as architecture configuration and existing system ADD. In addition, technologies and application types are commonly mentioned within the description. (e.g. “I am working on a RESTfull application”, “An existing process changes the status field of a booking record in a table, in response to user input.”, “I am using Apache MINA in my open source project”)

The table below list examples for the aforementioned defined ontology classes.

AK Ontology Class	Example
Design Issue	4741713 → “I want to send a batch of 20k JMS messages to a same queue. I'm splitting the task up using 10 threads, so each will be processing 2k messages. I don't need transactions.”
Requirement	4473567 → “Our criteria: 1. Short roundtrip time. 2. Low roundtriptime standard deviation. (We understand that garbage collection pauses and network usage spikes can affect this value). 3. High availability. 4. Scalability (we may want to have multiple instances of Ruby and Java app exchanging pointtopoint messages in the future). 5. Ease of debugging and profiling. 6. Good documentation and community support.”
Constrain	Team skills constrains: 13016406 → “I have never used Netty” Solution constrain: 12783677 → “This needs to adhere to WCF REST standards”
User Request	1582952 → “How do I choose between WCF, REST, POX and RIA services for a new Silverlight application”
Technology Feature	1429318 → “EMS is centralized (hub and spoke) on a specific server(s) and can traverse subnets no problem” 10375137 → “ActiveMQ is a widely used message broker that offers FIFO queues”
Technology ASTA	Benefit: 100993 → “It is much easier to debug Webservices over the wire as the data is SOAP/HTTP , which can be easily captured via sniffing tools for debugging” Drawback: 19758215 → “performance difference will be negligible and in many cases worse for NIO (Netty with thread sharing)”
Technology Use-case	12783677 → “An application I'm working on has a similar architecture, and I'm planning to use SignalR to push updates to clients, using long polling techniques (...) I have implemented this now, and it works very well”
Design	361491 → “I would highly recommend using WCF; and use the WCF

Decision	Service Library project over the Silverlight-enabled web service”
Decision Rule	17806977 → “If performance is your main criteria, you should definitely look at ZeroMQ.”

Simple ontology classes:

1. Technology name: represented with the name of a technology (e.g. “WCF”, “Netty”, “Biztalk”).
2. Technology type name: they represent a family of technologies (e.g. “Message queues”, “SOAP Library”, “message protocol”)
3. Pattern name: represented with the name of a pattern (e.g. “Messaging”, “Rest”, “FIFO” “Queuing”).
4. Quality attribute name: represented with the name of quality attributes based on ISO standard (Ref) (e.g. “Performance”, “Reliability”, “Interoperability”, “Scalability”)
5. Application types: They are usually represented with words “application” or “app” associated with an adjective to define the type of the application. (e.g. “distributed application“, “web app”, “Java application”, “Mobile app”).
6. Component name: As part of describing a software architecture. Processing units or storage components are mentioned using different terms. (e.g. “server”, “database”, “service”, “back end”, “system”, “client”, “process”)
7. Component element: They are elements which constitute a component. (e.g. “operation”, “method”, “job”, “event”, “interface”, “field”)
8. Connector verb/name: used to express communication to or from a certain component. They could be expressed using verbs as well as terms. (e.g. “send”, “receive”, “read”, “write”, “communication”, “connection”)
9. Connector data: they are the data need to be transferred through the connector to or from a component. (e.g. “data”, “request”, “response”, “message”, “object”)
10. Infrastructure term: they are terms used to describe an infrastructure or networking. (e.g. “firewall”, “internet”, “NAT”, “port”, “load balanced”, “data center”)
11. Cost: identified with words like “budget” or “cost”.
12. Programming activity: Describing common programming activities. (e.g. “debug”, “deploy”, “write code”)
13. Programming element: Describing common programming concepts. (e.g. “class”, “method”, “attribute”, “inheritance”, “query”)
14. Feature terms: Behavioral features are expressed through several technology component terms, as well as class names. (e.g. “Authenticator”, “SoapServer”, “serialize”, “endpoint”, “binding”, “socket”, “stream”). On the other hand, development features are expressed through programming elements, development tools, and programming activities.

Decision bound between ontology classes

1. Difference between technology solution features and requirements: Technology features describe the capabilities of the technologies. Usually verbs like “support” or “provide” are used to express that a certain technology offer a certain capability. On the other hand, requirements describe the needs of the user. Usually verbs like “require” or “need” or “would like” are used to express the wishes of users for certain quality attributes or technology features.
2. Difference between technology solutions use-cases and design decisions: Technology solution use cases describe a story in the past from the experience of the user. The story describes several context details. On the other hand, design decisions are presented as recommendations for using a certain technology solution using simple present tense (e.g. “use”, “go with”) as a response to the requirements and user request mentioned in the question section of the post.

3. Difference between design decisions and decision rules: Decision rules are a more complex structure than design decisions. Decision rules always have a conditional statement, while design decisions are coming in imperative or normal statements. Moreover, decision rules contain design decisions. However, design decisions could also come independently without decision rules.
4. Difference between technology solution features and benefits: even though technology solution features and benefits have similar structure. Technology benefits are differentiated through the extensive use of adjectives and adverbs (e.g. “very”, “fast”, “easy”, “better”), as well as the usage of keywords like “advantage” or “benefit”.

A.3.2 Atlas.ti Snapshots

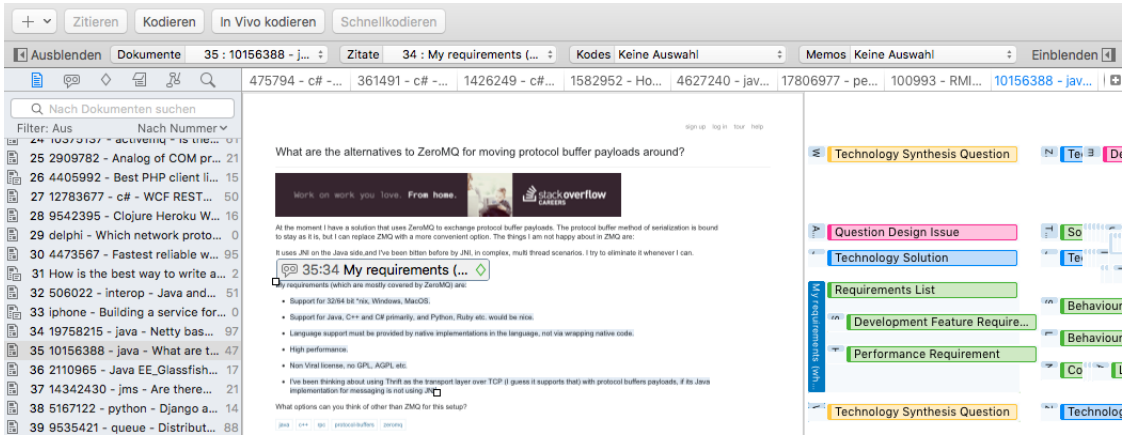


Figure A.2: Atlas.ti interface for documents coding

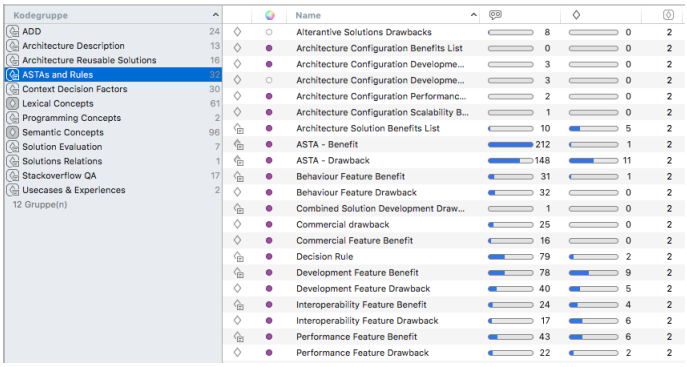


Figure A.3: Atlas.ti interface for managing codes

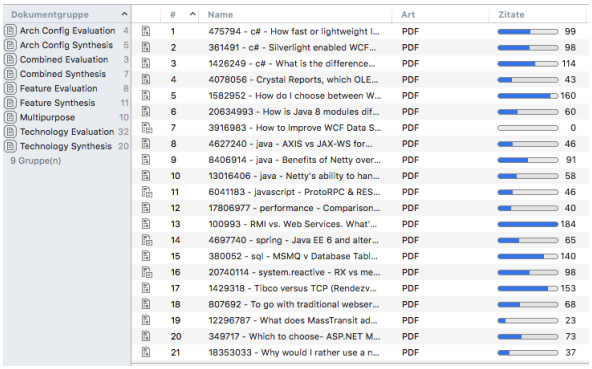


Figure A.4: Atlas.ti interface for managing documents

To analyze Stack Overflow posts for architecture knowledge concepts as have been explained in Chapter 5, we used the Atlas.ti[1] qualitative analysis tool. In this section, we present three snapshots for the tool and from our analysis. Fig. A.2 shows a snapshot for the Atlas.ti interface, where sentences are coded. At the left side is the list of documents, in the middle is the current coded document, and at the right side are the corresponding codes for each coded segment in a post. Fig. A.3 shows another interface from the Atlas.ti tool, where codes could be managed and organized. In this interface, we could specify categories of codes (at the left side), describe

[1] https://atlasti.com/

and understand codes (at the right side). Fig. A.4 shows the documents management interface of Atlas.ti. In this interface, posts could be categorized according to our categories in Chapter 4.

A.3.3 Instances of Ontology classes

In this subsection, we list additional examples for each of the ontology classes for our proposed ontology in Chapter 5. For the simple AK and lexical triggers ontology classes, we provide a comma separated list of words for each of the ontology classes. For the composite AK ontology classes, we provide several examples of sentences. Each sentence ends with the ID of the post, where the sentence belongs. To access the post, concatenate the ID of the post with the Stack Overflow URL (https://stackoverflow.com/questions/).

(TEC) Technology Solution (Examples, complete list in CD)
Protocolbuffers,Remoting,protobufnet,protobuf,protocol,protobufs,remoting,Silverlight,WebServices,BinaryFormatter,java,.NET,MTOM,Silverlight enabled WCF Service,Silverlight-enabled WCF services,WCF,web service,ADO.Net Data Service,ADO.NET Data Service,ADO.Net Data Services,ADO.NET Data Services,Silverlight,silverlight,Silverlight-enabled web service,Silverlight-enabled web service,Silverlight-enabled WCF service,silverlight-enabled wcf service,SOAP,REST,Entity Framework,Remoting,WSDL,SOAP,WS-I BasicProfile,.NET,SOAP,XML,CrystalReports,OLEDB,SQLServer,ODBC,SQL,ODBC,RIA,POX,WPF,AJAX,LINQ,JSON,OSGi,osgi,Jigsaw,CORBA,RMI,Axis,JAX,Webservice,Java,netty,ServerSocket,sockets,tcp,http,ApacheMINA,MINA,ByteBuffers,ProtoRPC,Protorpc,protorpc,Backbone,backbone,RabbitMQ,MSMQ,ZeroMQ,NServiceBus,MassTransit,RMI,HTTP
(PAT) Architecture Pattern
pattern,tactic,blocking,synchronous,sync,half-sync,half sync,asynchronous,async,half-async,half async,non-blocking,blocking,broadcast,messaging,broker,layer,client/server,client-server, client server,client,server,callback,looselycoupled,multicast,MVC,modalviewcontroller,MVP,MVVM,controller,view,NIO,messaging,queuing,queue,fifo,publishsubscribe,subscribe/publish,subscribe-publish,publish/subscribe,publish-subscribe,publish and subscribe,pub/sub,pub sub,pub-sub,broadcast,eventbased,eventbased,event/based,eventdriven,eventdriven,event/driven,REST,RESTful,routing,translator,translate,router,channel,endpoint,dispatcher,publisher,subscriber,rpc,remote procedure,remote-procedure,remote/procedure,poll,polling,peer to peer,peer-to-peer,peer2peer,p2p,supernode,service-oriented,service oriented,service/oriented,SOA,Software as a service,SaaS,store-and-forward queue,shared repository,shared-repository,active repository,activerepository,repository,blackboard,proxy,resourcepool,cache,lookup,evictor,handler,reactor,masterslave,master/slave,encapsulation,encapsulate,intermediary,cohesion,replication,interceptor,reflection,interpreter,microkernel,pipesandfilters,pipe,filter,facade,monitor,monitoring,ping,heartbeat,timestamp,sanity,voting,redundancy,shadow,orchestration,orchestrate,coherence,concurrency,authenticate,authentication,authorise,authorize,authorization,encrypt,encryption,ipc,loadbalancing,load balance,loadbalance,multitenant,multitenant,multitenant,named-pipes,namedpipes,named pipes,data-ingest,dataingest,transactional,stream,stream-storage,streamstorage,streamprocessing,streamprocessing,structured,unstructured,hot,warm,cold,batch,batch-processing,temperature,interactive analytics,interactive-analytics
(QA) Quality Attribute

accessibility,accessible,accountability,accountable,accuracy,accurate,adaptability,adaptable,adapte, administrability,administrable,affordability,affordable,auditability,auditable,availability,available,co mpatibility,compatible,composability,composable,configurability,configurable,correctness,credibilit y,credible,customizability,customizable,debuggability,debuggable,debatable,degradability,degradab le,determinability,determinable,demonstrability,demonstrable,dependability,dependable,depend,d eployability,deployable,deploy,distributability,distributable,distribute,durability,durable,effectivene ss,effective,efficiency,efficient,evolvability,evolve,evolvable,extensibility,extensible,fidelity,flexibility ,flexible,inspectability,inspectable,installability,installable,integrity,integrate,interchangeability,inter changeable,interoperability,interoperable,latency,learnability,maintainability,maintainable,mainten ance,manageability,manageable,mobility,modifiability,modifiable,modularity,operability,operable,o rthogonality,portability,portable,precision,predictability,predictable,producibility,provability,recover ability,recoverable,reliability,reliable,reliably,repeatability,repeatable,reproducibility,resilience,resp onsiveness,reusability,reusable,reuse,robustness,robust,roundtrip,safety,safe,scalability,scale,scalab le,scaling,seamlessness,sustainability,sustainable,serviceability,speed,supportability,securability,sec ure,security,simplicity,simple,stability,stable,survivability,tailorability,tailorable,tailor,testability,test able,throughput,traceability,traceable,transparency,transparent,ubiquity,understandability,underst able,upgradability,upgradable,usability,usable
(COM) Architecture Component
legacy,component,components,application,applications,machine,machines,server,servers,client,clie nts,users,user,backend,backends,service,services,system,systems,thread,threads,platform,platforms ,devices,device,source,sources,beans,bean,table,tables,record,records,queue,queues,front ends,front end,mainframe,host,hosts,APIs,API,app,apps,boxes,box,Q,physical locations,physical location,engine,engines,process,processes,program,programs,listener,Listeners,workers,worker,co mputers,computer,model,models,entities,entity,datastore,datastore,store,context,ViewModels,dat abase,producers,producer,consumers,consumer,session,sessions,logic,page
(CON) Architecture Connector
retrieve,retrieves,retrieving,commit,commits,committing,consuming,consume,consumes,communic ation,communications,communicate,communicated,communicates,execute,executes,execution,con nect,connected,connects,connects,connection,connectivity,write,writes,writing,upload,uploading,u ploaded,workwith,send,sends,sending,sent,implementing,implement,implements,store,stores,stori ng,forward,forwards,forwarding,check,checking,checks,changes,change,stream,streams,streaming,r eceive,receives,received,receiving,deliver,delivers,delivering,call,calls,calling,called,talkwith,expose, exposes,exposing,accessing,access,accessed,interacting,interact,interacts,interaction,fedfrom,read,r eads,reading,dumps,dump,dumping,grab,grabs,pushing,push,link,links,linking,routing,route,share,s hares,shared,sharing,gets,get,getting,pull,pulls,pulling,collect,collects,collecting,save,saving,saves,ac cept,accepting,accespts,queries,transmit,transmits,transmitted,transmitting,exposing,exposes,expo sed,trigger,triggers,triggering,triggered,notify,notifies,notified,notifing
(COME) Component Element
method,methods,stored procedure,procedure,procedures,operation,operations,interface, interfaces,field,fields,event,events,button,buttons,text box,endpoint,endpoints,job,jobs,function,functions,class,classes,file,files,module,modules
(COND) Connector Data
socket,data,payload,payloads,object,objects,messages,message,XML,dump,updates,requests, request,map ,structure,structures,map,maps,reply,replies,item ,items,list,lists,result,results,tasks,task,information,call,calls,kb ,mb,data,notification

(PROB) Software Problem
SPOF, point of failure, road block, error, out of memory, lock, contention, heavily loaded, take long
(FT) Feature Term (Examples, complete list in CD)
BinaryFormatter (remoting),serialize/deserialize,serialize arbitrary types,protocol buffers wire format,injection,contract,contract-first,queries dynamically,query and data changes back to the server,execute the method on the remote machine,cross platform,talks in a binary format,Web service interoperability,create an object,send a message,work through firewalls,platform independent,programming independent,binary serialization,generates classes,Authenticator class,Authenticator,sockets,pipelines, decoders,streams,asynchronous request,CRUD,service discovery,can call session beans via RMI,transfered in a binary format (usually),the data is SOAP/HTTP,interface specification is explicitly stated,EJB container,expose them as web services,Service Broker,an independant processing executable,offload the work to another server,asynchronous server queries,use and combine different Events,decentralized: there is an rv client on every host,middleware application designer,client acknowledgment,centralized (hub and spoke) on a specific server,can traverse subnets,subjects (or JMS topics),connectors,MVC controllers,WCF Data Services/ODATA,platform invoke,BAPIs and function modules
(ACTV) Programming Activity
acquire,activate,add,adjust,align,append,apply,archive,assign,attach,bind,book,boost, bootstrapp,break,cache,calculate,catch,center,checkout,clean,clone,compile,complete,compose,compress,compute,configure,convert,count,create,crop,customize,deactivate,debug,defer,define,delete,deploy,describe,determine,develop,disable,discontinue,display,divide,document,download,duplicate,edit,embed,enable,encapsulate,encode,enter,exclude,expand,extend,fetch,fill,find,fix,flush,force,format,generate,group,handle,hardcode,hide,host,identify,ignore,import,include,indent,inherit,initialize,insert,install,instantiate,integrate,introduce,isolate,learn,limit,list,locate,log,login,login,manage,manipulate,mark,match,mock,modify,move,multiply,obtain,omit,open,order,output,override,overwrite,pass,paste,patch,perform,place,prefix,prepare,present,prevent,price,print,process,provide,purchase,raise,reach,rebuild,recompile,recreate,redefine,redirect,refactor,reference,regenerate,release,remove,rename,render,repeat,replace,replicate,request,resolve,restart,return,sell,separate,seperate,set,setup,shorten,show,sign,simulate,skip,sort,specify,split,submit,summarize,surround,switch,test,throw,track,translate,trim,update,wrap
(ADV) Advise Verbs
recommend, favor, suggest, propose, commend, advise, advice, encourage, motivate
(AMT) Amount Term
many, multiple, thousands, hundreds, tons, all, lots, much, hundreds, dozens, heaps, piles
(ASES) Assessment Verbs
evaluate, assess, appraise
(BHV) Behavioral Verbs
run, perform, process, listen
(CHR) Charachtersitc Nouns
pros, cons, advantage, disadvantage, benefit, drawback, strength, weakness, deficiency, flaw, fragility, shortcoming, prononess, value, favor, favour, positive, negative
(CONC) Concern Nouns
concern, requirement, demand, request, care, worry, essential, necessity, requisite,

(CONS) Constraint Nouns
limitation, constraint, restriction, restraint
(DFF) Difference Noun
difference, dissimilarity, distinctness, distinction, compare
(DIF) Difficulty Adjectives
difficult, hard, overkill, nightmare, unmanageable, arduous, backbreaking, grueling, gruelling, heavy, laborious, operose, punishing, toilsome, tough, overhead, effort, stress, much work
(DIS) Discover Verbs
search, look, seek, research, explore, consider
(FIT) Fit Verbs
fit, hook, play, complement, integrate, work, incorporate, run
(FORC) Force Verbs
have to, forced, must
(LER) Learn Verbs
learn, acquire, larn, study, teach
(PROB) Problem Nouns
problem, trouble, issue, snag, obstacle, hurdle
(QUE) Question Word
which, what, when, how
(REL) Rely Verbs
depend, rely, implement, count on, build
(SOL) Solution Adjectives
alternative, option, choice, direction, way, trend, solution, tendency
(SPED) Adjectives
fast, quick, rapid, robust, efficient
(STAY) Stay Verbs
stick, adhere, avoid, stay, bind, bond, impel, avert, evade
(SUPP) Support Verbs
support, provide, offer, supply, allow
(USE) Use Verbs
take, prefer, favor, opt, go, use, utilize, utilise, choose, select, pick
(VAL) Value Adjectives
good, fine, well, nice, superb, cool, brilliant, magnificent, top
(VS) Versus Preposition
versus, vs, against, opposition, opposed, contrast, contrary, opposing, counter, differ
(WISH) Wish Verbs
need, require, want, demand, ask, like, wish, hope, plan, desire
(CONF) Architecture configuration (Examples, complete list in CD)
retrieve/commit some data from/to a database back on the server (361491) execute the method on the remote machine (1426249) Silverlight application can connect back to it' server (1582952) connect multiple users to your back end (4627240)

pass the new client Socket off to some processing code in a new thread (8406914)
a platform that would be able to stream large chunks of data over HTTP (13016406)
send complex object nets from one application to another (100993)
Receiving a message from a distributed queue (20740114)
deliver things such as stock quotes to our front end (1429318)
accessing some internal data from a production server (2198168)
C# .NET Winform clients interacting with Java/JBoss middle-tier via Tibco EMS messaging (37579)
Clojure (Java) and Ruby apps to communicate (4473567)
it gets one item of a list and does some math and uploads the result to a database (9535421)
Application consist of two component Web application and Windows .NET Application. (2249715)
(CB) Component Behavior (Examples, complete list in CD)
service is simply a front-end for the database (1582952)
terminate once the processing is complete and the response is sent. (8406914)
process to write, that will run asynchronously for records with a particular status (380052)
service need to be available over multiple communication protocols (807692)
60 boxes running each 10 clients all do task X (37579)
Java Queue that will be automatically distributed across the whole cluster (9535421)
Domain classes are a nice place to put business logic including complex validations (2897513)
Its responsibility is to scan all active network, find all IT assets (2249715)
The API call is executed by the broker (4741713)
(EX) Existing System (Examples, complete list in CD)
I've built a web app, and a Silverlight control (361491)
I am using Apache MINA in my open source project (13016406)
An existing process changes the status field of a booking record in a table (380052)
contention with two process reading and writing to the same region of the bookings table (380052)
At the moment I have a solution that uses ZeroMQ to exchange protocol buffer payloads (10156388)
(DI) Design Issue (Examples, complete list in CD)
I've built a web app, and a Silverlight control. I will be adding one of those 3 options to my web application and consuming it from my Silverlight component (361491)
I'm doing something like simulation where I need to generate a lot of things on the smaller machines seeded by a value from a database, but these are reduced before they return to the source machine/database. (18521196)
I'm thinking about using web sockets with Netty for an application where clients connect to a server to get some information at first. Then, they are registered by the server and any changes on the information of a particular client will trigger a notification to the client containing the updated information. In this case, the communication is first initiated by the client and is latter initiated by the server (12054412)
The system we are building is receiving data through the external feed. Our job is to distribute this data to multiple services, run the calculations and forward the results elsewhere - typical publisher-subscriber situation. (2124221)
I'm building native mobile applications in both iOS and Android. These apps require "realtime" updates from and to the server, same as any other network-based application does (Facebook, Twitter, social games like Words with Friends, etc) (6614343)

I have a WCF layer and my Domain Model is behind this WCF layer. I am using Nhibernate as an ORM tool and all my business logic/ Data Access etc will be behind this WCF layer. I am exposing DTO to my clients. (20091826)
I may need two distinct patterns/practices/strategies:1. Loading a large number of records over the internet (~5k).2. Keeping a subset of these objects (~500) update-to-date over the internet. (1670332)
(REQ) Requirements and Constraints (Examples, complete list in CD)
web client app has a need to connect multiple users to your back end simultaneously (4627240)
system needs to be able to handle many (thousands) of connections at the same time (8406914)
need to be decoupled (100993)
high availability is important (37579)
Short round-trip time.2. Low round-trip-time standard deviation (4473567)
Changing or reworking the UI should be as easy as possible (13592894)
What we need is a very low latency messaging. (2124221)
clients are not under your control or might move to another platform (100993)
have a limitation on time on deliverable. (807692)
The client-side for this is being written in Silverlight. (1670332)
The protocol buffer method of serialization is bound to stay as it is (10156388)
(UR) User Request (Examples, complete list in CD)
Do they differ much in how difficult they are to implement? (2576446)
Are Websockets adapted to very long-lived connections? (12054412)
why wouldn't couchdb be a good platform to deliver such updates? (2912051)
Is RabbitMq fast enough for a soft realtime message delivery? Are there any benchmarks? (2124221)
Did you have any trouble installing? Is it stable?Were there any performance issues?
How is the documentation / support? (4405992)
how RX differs from messaging queues like RabbitMQ or ZeroMQ? (20740114)
What, if any, are the benefits of Netty for such simple requirements? (8406914)
How do I choose between WCF, REST, POX and RIA services for a new Silverlight application (1582952)
when one is appropriate to use over another. What pros/cons do each offer? (361491)
(FEAT) Technology Feature (Examples, complete list in CD)
ZeroMQ / RabbitMQ solutions often have to use and combine different Events quite a bit, which Rx is very good at (20740114)
JMS an external system just like the database. JMS message are received using special kind of EJB called Message-driven bean (MDB). (2110965)
Oledb (Object Linking and Embedding DB) is a standard format supported by a large number of dbs, so you can connect to oracle, db2 etc (3766433)
Camel does allow you to integrate with XMPP so that you can consume messages from or produce to such a mechanism (10051261)
protobuf-net (Marc's) is a ground-up re-implementation following the same binary format (indeed, a critical requirement is that you can interchange data between different formats), but using typical .NET idioms:mutable data classes (no builders)the serialization member specifics are expressed in attributes (comparable toXmlSerializer , DataContractSerializer , etc) (475794)

Web services are cross platform, using common standards and work through firewalls. They also think in terms of messages, not objects - you send a message to a service, and you get a reply (1426249)
BSON, ProtoBuffers, and BERT offer serialization of arbitrary data structures (numbers, strings, sequential arrays, associative arrays) into binary values.(4473567)
these technologies have the APIs to access the native Devices' resources like Battery check, SMS, MMS, GSM broadcast channels, Contacts, Lighting, GPS , and Memory (6614343)
WCF gives you a level of abstraction over the way you are/want to communicate. So, you can choose binding that is Microsoft-specific, but you can also use SOAP protocol, or, you use both, so non-Microsoft client will be able to communicate through fe. SOAP, and other client can use more robust ways. (1262415)
RabbitMQ is a sophisticated server product that provides complex messaging patterns, topics and routing out-of-the-box. (17806977)
NServiceBus which implements a complete Service Bus architecture, and could be used in a ServerAndClient (3026345)
(ASTA) Technology Benefits and Drawbacks (Examples, complete list in CD)
My code tends to be slightly faster than his, but both are much, much faster than the other serialization/deserialization options in the framework. (475794)
you should get query and update very cheaply with this approach (361491)
It doesn't work well with firewalls (1426249)
RMI doesn't deal with guaranteed delivery or asynchronous responses (2576446)
building REST application would be simpler and faster (2751752)
where XMPP really shines is in its extensibility (10051261)
Netty also offers a very nice API for building a server. (5145129)
An ESB tends to provide more than one communication interface. (2023130)
(ADD) Recommended Design Decisions (Examples, complete list in CD)
I suggest you try Mule for development and WebSphere ESB for test and production. (2023130)
Erlang is very well suited for your use case. (6614343)
You can check out some more advanced js libraries like KendoUI (9031116)
you'd need session and connection per-thread due to the JMS thread model (4741713)
I would consider moving away from the relation database in favor of for example Cassandra. (2567254)
Have you considered something like Storm or Spread? (10156388)
(DR) Decision Rule (Examples, complete list in CD)
If Performance - whatever you construe it to mean - really is your #1 criterion, then you should probably abandon SOAP and POX and move to protobufs or something else optimized for performance. (2751752)
If you want flexibility in running workflowed tasks use Celery. (18521196)
If you are talking about an HTTP web application, go with the tried an true. Apache for straight HTML pages, Tomcat if you need Servlets. (5145129)
If you want your connection to stay alive continuously for long periods then I would suggest adding some logic to your client to reconnect when the onclose event happens (12054412)
You would use one when you need to reliably send a inter-process/cross-platform/cross- application message that isn't time dependent. (2868800)

If your intentions are to simply build a DAL in a typical N-Tier fashion, there is no reason that a simple class library wouldn't suffice. (401904)
If you are doing fast things with small data sizes then go with something homegrown and leverage a TCP connection. (6614343)
If the sessions are transacted then you'd need session and connection per-thread due to the JMS thread model. (4741713)
if you have to deal with multiple development languages, using an interface running over HTTP is a very pragmatic solution (2567254)
If high availability is important Amazon SQS is worth looking at. (37579)
if you're building a heterogeneous distributed application involving messaging
code running on Windows, Unix (AIX/Solaris), Linux, or Mac OS X, then Tibco EMS is the ticket (37579)
if you have a PI (XI) system you could build an interface via PI and expose that as a webservice to the .NET world. (2198168)

(CASE) Technology Use-case (Examples, complete list in CD)

The Sipdroid developers found that persistent TCP connections do greatly improve battery life. Their article on the topic doesn't address the server side but it does give a high-level description of their approach on the client. (6614343)
The Evolution email client as part of GNOME uses CORBA. It uses ORBit (2909782)
Long-lived connections is what WebSocket was designed for. Depending on how your clients connect, those connections might nevertheless be limited in lifetime, i.e. on retail DSL connections, there often is a forced reconnect every 24h at least. (12054412)
Imagine that you want to notify a bank system of ATM money withdrawal, and it has to be done exactly once per request, no matter what servers failed temporarily in the middle. MQ systems would allow you to coordinate transactions across multiple database, MQ and other systems. (2868800)
In summary Data Services are designed to be client facing, you are exposing your data so it can be accessed over the web from some other body. While you could force data services to fit into a back-end server data access layer, you should only do so if you can find a justifiable reason to do so (2508361)
A classic example is a stock price ticker service - this might be delivered via message queuing, but then transformed by Rx to group, aggregate and filter prices (20740114)
We're running RabbitMQ in EC2 with persistence for all the messages. We use only m1.large instances (64bit with high IO performance). We started with EBS storage, then switched to instance-store, to see if there's any improvement. And instance-store instances are faster in terms of IO throughput. But, the drawback is that all persisted messages are lost along with the termination/failure of the instance (although we never experienced a failure ever, so far). In our scenario, we don't need such a big throughput, but we do care a lot if our messages get lost (6939110)
Our team is in the process of implementing a Silverlight app on top of the RIA stack. We've decided to build a domain model on top of the RIA entities. Additionally, we elected to follow the MVVM pattern to model UI interactions. (2897513)

B

Interviews Materials

B.1 Interview Questions to Understand Technology Design Decisions 204
B.2 Interview Questions to Understand the Use of Developer Communities . . 205

B.1 Interview Questions to Understand Technology Design Decisions

We asked the following questions during our interviews to understand how practitioners take technology design decisions (Answers RQ1, see Chapter 3).

Phase 1 - Participants background:

1. How long is your experience with software technologies?
2. Which technologies do you consider yourself expert in?
3. What type of projects you worked on? The domain of business, and how big are the systems?
4. Which roles you worked in during your work? Currently and previously?
5. Have you took a technology architecture decision before? How often you take these decisions?
6. What does software architecture mean to you?

Phase 2 - Personal Experience:

1. From your experience, are technology decisions important design decisions for the architecture of a software system?
2. Do you consider selecting conceptual solutions such as patterns important? and how are they related to technologies?
3. If you have two technologies, what makes an architect choose a technology over the others?
4. How would you decide on a solution, if you have two technologies, and both solve the same problem?
5. Is it then important to know the consequences of the decision?

6. Can an architect know all details of a technology?
7. What is important to know about a technology?
8. Sometimes technologies are based on patterns, is it considered in design decision?
9. Does the development environment influence the decision of selecting a technology?
10. Does usability features of a technology influence the decision to select a technology?
11. Is operational aspects, such as monitoring tools of a technology influence the decision of selecting a technology?
12. Do run-time features such as performance influence a technology decision more than development features?
13. If you have list of technologies, and you would like to choose from them? What are the steps? If we assume, there is no constrains?
14. If you are forced to select a technology, What would you do?

B.2 Interview Questions to Understand the Use of Developer Communities

We asked the following questions during our interviews to understand how practitioners use developer communities to find architectural knowledge (Answers RQ6, see Chapter 6).

Phase 1: Participants background:

1. How many years of experience do you have with software technologies and development? And software architecture?
2. Which technologies you consider yourself expert in?
3. In which domain/industry (e.g. Finance, Telecom) do you work in?
4. How big is your organization?
5. Which roles you worked in during your work? Currently and previously?
6. What does software architecture mean to you?
7. Which software tools do you use during software design?
8. Did you consult developers community (e.g., internet blogs, forums, Stack Overflow etc.) before during architecture design?

Phase 2 - Current use of architecture knowledge in developer's communities:

1. Do you consult other information sources (other than your personal experience) when making design decisions? What are they?

2. What are the advantages of consulting developers community (e.g., internet blogs, forums, Stack Overflow etc.) when making design decisions?
3. Which developer community websites do you consider during your searching? How do you choose between them? Give an example.
4. What type of information are you usually searching for in developers community? At which phase in the design do you use it? For which design decisions? Give an example.
5. How do you specify your searching criteria? Which information do you consider for searching?

Phase 3 - Identify concerns related to searching for AK, ideal approach for AK reuse from developers' community:

1. Do you find it easy to find the information you need across distributed developers community pages? Why?
2. Which information are easy/hard to find in developers community?
3. What are the problems, which face you during searching for AK in developersâĂŹ community? What do you propose for a solution to overcome these problems?
4. How would you envision an approach that helps in search for and reuse architecture knowledge in developers community?

Wrap-up Upon reflection, after answering the questions, Is there anything you can add and that you feel is relevant in the context of this interview?

After conducting the interviews with practitioners in Chapter 6, we asked them to fill a survey, which is included in the following 3 pages.

Goal of the survey:
The main goal of the survey is to determine the importance of the different scenarios for accessing architecture knowledge in developer communities.

Terms Definition:

1) *Community or developer communities* are web pages which provide the possibility for technology experts and architects to communicate with each other and exchange knowledge. For example, Blogs, Forums, Stackoverflow, yammer. These communities could be offered from private companies (e.g. Stackoverflow), or from technology vendors (e.g. Microsoft), or personal web sites (e.g. Blogs).

2) *Architecture components design*: It's a representation of the components of the system and how they interact with each other. The modeling of system components could be done in different views (static vs. dynamic), and using different notations (e.g. UML).

Searching in community webpages

1) How often do you search for suitable technologies (e.g. frameworks) or patterns in community web pages, in order to satisfy requirements or solve a design problem?

   ☐ Often (for most design problems)
   ☐ Sometimes (for many design problems)
   ☐ Rarely (for few design problems)
   ☐ Never

2) How often do you search for information about evaluations or comparisons between technologies (e.g. frameworks) in community web pages?

   ☐ Often (for most design problems)
   ☐ Sometimes (for many design problems)
   ☐ Rarely (for few design problems)
   ☐ Never

3) „*After choosing a technology, you need to design the right components of the system, taking into consideration the features of the selected technologies*."
   How often do you search for information about technology features in community web pages to help developing an architecture component design of a system?

   ☐ Often (for most design problems)
   ☐ Sometimes (for many design problems)
   ☐ Rarely (for few design problems)
   ☐ Never

4) „*After developing the components design of the system*" How often do you search for similar architectures in developers community, in order to evaluate your proposed design?

   ☐ Often (for most design problems)
   ☐ Sometimes (for many design problems)
   ☐ Rarely (for few design problems)
   ☐ Never

5) „*You write searching keywords to search for certain architecture information, and get a list of relevant webpages*“. How hard is to read and classify webpages in order to find relevant architectural information?

☐ Complex and time consuming
☐ Easy / Doable
☐ Trivial (Not a problem)

6) „*After writing the searching keywords to search for relevant architecture knowledge, and getting a list of relevant webpages*“. For a typical design problem, how do you read and analyse community webpages for the required knowledge?

☐ Read each word and sentence to completely understand the content of the page.
☐ Search for certain words and sentences, which contain the needed architecture knowledge.

7) „*During searching for architecture knowledge, you read several webpages to find the information you need*“. How often do you check the validity of the information in community webpages?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)
☐ Rarely (for few design problems)
☐ Never

Searching during modeling and documentation

8) How often do you search for architecture information *in between or during modeling and documenting an architecture design*?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)
☐ Rarely (for few design problems)
☐ Never

9) In searching for architecture knowledge during modeling and documentation, do you consider the terms used in architecture models and requirement documents as keywords to search for architecture knowledge?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)
☐ Rarely (for few design problems)
☐ Never

Submiting problems

10) How often do you submit a question about a design problem to one of the developers communities?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)

☐ Rarely (for few design problems)
☐ Never

11) Which types of questions related to design problems do you submit to developers communities?

☐ Search for a suitable technology (e.g. framework, database,...)
☐ Evaluate a technology regarding its features or quality (e.g. performance, security, scalability).
☐ Ask for additional information about technology features
☐ Ask about suitable architecture components design
☐ Ask to evaluate a complete architecture components design

Browse architectural solutions features

12) How often do you browse technology vendor websites and developer communities (e.g. stackoverflow) to understand its technology features?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)
☐ Rarely (for few design problems)
☐ Never

13) During which architecture design activity do you browse technology features?

☐ During searching for a suitable technology (e.g. Framework)
☐ During evaluating technology regarding its features or quality (e.g. performance)
☐ During asking for suitable architecture components design
☐ During asking to evaluate a complete architecture components design

Browse and contact architecture and technology experts in communities

14) How often do you use communities to find and contact technology experts, who could provide information to evaluate your design?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)
☐ Rarely (for few design problems)
☐ Never

15) How often do you use communities to find and contact technology experts, who could provide information about technology products and their architecture?

☐ Often (for most design problems)
☐ Sometimes (for many design problems)
☐ Rarely (for few design problems)
☐ Never

16) For which architecture design activity do you contact experts:

☐ During searching for a suitable technology (e.g. Framework)

☐ During evaluating technology regarding its features or quality (e.g. performance)
☐ During asking for suitable architecture components design
☐ During asking to evaluate a complete architecture components design

C

Experiments Materials and Detailed Results

C.1 List of Stack Overflow Tags 211
C.2 List of Distinctive Keywords and Ontology Classes 215
C.3 Experiment to Search for Architectural Information in Stack Overflow . . 225

C.1 List of Stack Overflow Tags

In Chapter 7, we conducted an analysis to explore if tags could be enough to identify architecture posts. As part of the analysis, we list tags, which have the highest differences in their occurrences between architecture and programming posts. We list here the top 100 tags, a complete list for tags is available in the attached CD.

Tag	ARP Percentage	Non-ARP Percentage	Difference
web-services	11.39534884	4.924623116	6.470725722
soap	9.186046512	3.115577889	6.070468622
message-queue	6.046511628	0.904522613	5.141989015
rest	6.860465116	2.010050251	4.850414865
java	15.81395349	11.15577889	4.658174594
rabbitmq	8.372093023	4.120603015	4.251490008
jms	6.046511628	1.909547739	4.136963889
zeromq	4.302325581	0.502512563	3.799813019
rpc	4.418604651	0.804020101	3.614584551
soa	3.720930233	0.301507538	3.419422695
architecture	3.488372093	0.100502513	3.38786958
messaging	3.720930233	0.40201005	3.318920182
python	6.395348837	3.417085427	2.97826341
esb	2.790697674	0.301507538	2.489190137
msmq	3.139534884	0.804020101	2.335514783
amqp	5.23255814	3.015075377	2.217482763
performance	2.558139535	0.502512563	2.055626972
asp.net-mvc	2.441860465	0.502512563	1.939347902
thrift	2.209302326	0.502512563	1.706789763
redis	1.511627907	0.100502513	1.411125394
netty	1.744186047	0.40201005	1.342175996
amazon-web-services	1.627906977	0.301507538	1.326399439
json	3.255813953	2.010050251	1.245763702
gwt	2.093023256	0.904522613	1.188500643
entity-framework	1.279069767	0.100502513	1.178567255
node.js	1.627906977	0.502512563	1.125394414
authentication	1.162790698	0.100502513	1.062288185
sockets	1.162790698	0.100502513	1.062288185
publish-subscribe	1.046511628	0	1.046511628
javascript	1.744186047	0.703517588	1.040668459
android	2.093023256	1.105527638	0.987495618
ipc	1.046511628	0.100502513	0.946009115
design-patterns	0.930232558	0	0.930232558
security	1.511627907	0.603015075	0.908612832
wsdl	2.209302326	1.306532663	0.902769662
api	1.162790698	0.301507538	0.86128316
queue	1.162790698	0.301507538	0.86128316
protocol-buffers	1.046511628	0.201005025	0.845506603
websocket	1.046511628	0.201005025	0.845506603
jax-ws	1.046511628	0.201005025	0.845506603
web-applications	0.813953488	0	0.813953488
asynchronous	1.395348837	0.603015075	0.792333762
wcf-data-services	1.279069767	0.502512563	0.776557205
scala	1.162790698	0.40201005	0.760780647

celery	3.372093023	2.613065327	0.759027697
ajax	1.046511628	0.301507538	0.74500409
wcf-ria-services	1.627906977	0.904522613	0.723384364
distributed	0.813953488	0.100502513	0.713450976
objective-c	0.697674419	0	0.697674419
xmpp	0.697674419	0	0.697674419
erlang	0.697674419	0	0.697674419
mqtt	0.697674419	0	0.697674419
nio	0.697674419	0	0.697674419
twisted	0.697674419	0	0.697674419
poco	0.697674419	0	0.697674419
networking	1.046511628	0.40201005	0.644501578
iphone	1.046511628	0.40201005	0.644501578
c	1.046511628	0.40201005	0.644501578
ios	0.813953488	0.201005025	0.612948463
wcf-web-api	0.813953488	0.201005025	0.612948463
client-server	0.697674419	0.100502513	0.597171906
servicebus	0.697674419	0.100502513	0.597171906
asp.net-web-api	0.697674419	0.100502513	0.597171906
tibco	0.581395349	0	0.581395349
protocols	0.581395349	0	0.581395349
ibm-mq	0.581395349	0	0.581395349
nservicebus	0.813953488	0.301507538	0.512445951
django	1.511627907	1.005025126	0.506602781
rmi	1.511627907	1.005025126	0.506602781
azure	0.697674419	0.201005025	0.496669393
mule	0.697674419	0.201005025	0.496669393
tcp	0.697674419	0.201005025	0.496669393
asmx	0.697674419	0.201005025	0.496669393
remoting	1.395348837	0.904522613	0.490826224
duplex	0.581395349	0.100502513	0.480892836
nettcpbinding	0.581395349	0.100502513	0.480892836
distributed-computing	0.465116279	0	0.465116279
json-rpc	0.465116279	0	0.465116279
frameworks	0.465116279	0	0.465116279
cross-platform	0.465116279	0	0.465116279
actor	0.465116279	0	0.465116279
domain-driven-design	0.465116279	0	0.465116279
comet	0.465116279	0	0.465116279
gwt-rpc	1.162790698	0.703517588	0.45927311
wcf-binding	1.046511628	0.603015075	0.443496553
linux	0.930232558	0.502512563	0.427719995
windows-services	0.697674419	0.301507538	0.396166881
servicemix	0.581395349	0.201005025	0.380390324
amazon-ec2	0.465116279	0.100502513	0.364613767

integration	0.465116279	0.100502513	0.364613767
scalability	0.465116279	0.100502513	0.364613767
wshttpbinding	0.465116279	0.100502513	0.364613767
design	0.465116279	0.100502513	0.364613767
push-notification	0.465116279	0.100502513	0.364613767
akka	0.465116279	0.100502513	0.364613767
java-ee	1.162790698	0.804020101	0.358770597
unix	0.348837209	0	0.348837209
mongodb	0.348837209	0	0.348837209
haskell	0.348837209	0	0.348837209

C.2 List of Distinctive Keywords and Ontology Classes

In Chapter 7, we conducted an analysis to explore the most distinctive keywords and ontology classes to identify and classify architecture relevant posts. We list in the next 3 pages the top 100 keywords with the highest information gain ratio. A complete list for keywords is available in the attached CD. After presenting the top keywords, we present the complete list of the most distinctive ontology classes. Moreover, we present the top 200 distinctive sequences of ontology classes. A complete list of sequences are available in the attached CD. The lists of ontology classes and sequences use the same IDs of ontology classes as proposed in Chapter 5.

Terms in Title (Information Gain, frequency, term)		Terms in Question (Information Gain, frequency, term)		Terms in Answers (Information Gain, frequency, term)	
0.17082	1780 soa	0.17627	1266 scalability	0.17651	1274 throughput
0.16885	1039 alternatives	0.17452	1064 compared	0.16834	1235 scaling
0.16679	1894 versus	0.16462	1327 wed	0.16233	1304 xmpp
0.15143	1159 comparison	0.16233	1234 pros/cons	0.15901	1040 cloud
0.15143	1467 lightweight	0.16233	1289 subscribers	0.15487	1233 scalable
0.15143	1569 notification	0.15901	1233 pros	0.14798	1275 tibco
0.15143	1125 choosing	0.15901	1078 cons	0.14076	1030 broker
0.14806	1240 distributed	0.15449	1183 meet	0.14017	1098 esb
0.14806	1064 apps	0.15143	1248 real-world	0.1401	1093 enterprise
0.14426	1682 real-time	0.1513	1019 amqp	0.13988	1114 governance
0.14426	1085 backend	0.14806	1335 xmpp	0.1373	596 messaging
0.14426	1584 oriented	0.14806	1196 mule	0.13609	1031 brokers
0.14426	1641 pros	0.12403	1089 decision	0.13603	1217 redis
0.14426	1681 ready	0.11647	1277 soa	0.13202	1246 soa
0.14426	1508 middleware	0.11589	1084 corba	0.12884	1142 lightweight
0.14426	1175 cons	0.11093	1186 messaging	0.12864	1232 scalability
0.13988	1767 share	0.10624	1267 scalable	0.12838	1284 udp
0.13988	1294 experience	0.10624	1030 balancing	0.12767	1166 mule
0.13988	1249 dto	0.10624	1026 availability	0.12731	1000 activemq
0.13988	1154 communicate	0.10624	1103 dtos	0.12575	1034 bus
0.13988	1100 broker	0.10427	1018 alternatives	0.12528	1032 buffers
0.13988	1456 layer	0.10211	1014 advantages	0.12437	1234 scale
0.13988	1455 latency	0.10178	1032 benefits	0.12403	1276 tier
0.13731	1506 messaging	0.10066	1188 middleware	0.12403	1091 ems
0.13463	1694 rendezvous	0.10066	1106 ec2	0.12186	1264 systems
0.13463	1236 disadvantages	0.10002	1040 broker	0.12105	1106 faster
0.13463	1219 delivery	0.09441	1160 ipc	0.11966	1261 subscribers
0.13463	1938 xmpp	0.09441	1314 twisted	0.11966	1172 nservicebus
0.13463	1089 bean	0.09152	1298 technologies	0.11848	1153 mature
0.13463	1598 payloads	0.08827	1117 entities	0.11806	1026 benefits
0.13463	1387 implemented	0.08764	1081 considering	0.11768	1266 technologies
0.13463	1104 business	0.08737	1168 latency	0.11609	1062 coupled
0.13463	1734 scalable	0.08732	1260 round	0.11496	1012 amqp
0.13463	1098 bridge	0.08557	363 error	0.11129	1242 servicemix
0.13463	1111 cases	0.08484	143 architecture	0.11109	984 //activemq
0.13463	1769 sharing	0.08387	1119 esb	0.11096	1267 technology
0.13463	1874 udp	0.08387	1268 scale	0.11019	694 protocol
0.13463	1095 boost	0.08312	883 trace	0.1095	1202 producer
0.13463	1736 scale	0.08308	1069 complexity	0.10907	1204 protocols
0.13463	1106 c/c++	0.08308	1171 lightweight	0.1067	27 </code></pre>
0.13463	1773 similar	0.08198	374 exception	0.10585	1028 biztalk
0.13429	1045 amqp	0.08163	978 }	0.10521	1069 delivery

0.12787	1613 platforms	0.08056	1101 distributed	0.10509	1230 routing
0.12787	1292 exchanges	0.08049	1136 future	0.10464	1273 thrift
0.12787	1257 ecommerce	0.07949	974 xp	0.10386	1127 integration
0.12787	1856 topics	0.07917	1307 tier	0.10351	1126 integrate
0.12787	1862 transactional	0.07917	1016 akka	0.10294	1165 mq
0.12787	1674 rapidly	0.07908	1295 systems	0.10178	1305 zeromq
0.12787	1858 tornado	0.0785	1284 stick	0.10178	1060 corba
0.12787	1848 throughput	0.07836	849 svc	0.10128	1281 transactions
0.12787	1778 sns	0.07777	590 namespace	0.10002	1189 persistence
0.12787	1853 toolkit	0.0774	1200 notifications	0.09957	1079 distributed
0.12787	1861 traditional	0.0774	1065 comparing	0.09918	1004 advantages
0.12787	1830 swing	0.0764	1049 choose	0.09748	1015 architecture
0.12787	1386 implementations	0.0764	1048 choice	0.09714	1155 middleware
0.12787	1214 dealing	0.07639	1123 experiences	0.09564	1101 expensive
0.12787	1492 managing	0.07639	1082 consumers	0.09543	1225 requirements
0.12787	1661 pyro	0.0759	158 attempting	0.09437	1191 personally
0.12787	1114 celerybeat	0.07526	299 debug	0.09315	1303 xml-rpc
0.12787	1205 daemon	0.07512	1167 languages	0.09313	709 rabbitmq
0.12787	1401 inter	0.07512	1192 mostly	0.09268	1049 complexity
0.12787	1645 protobuf	0.07314	164 automation	0.09217	1228 robust
0.12787	1407 interprocess	0.07314	352 empty	0.09121	1063 curve
0.12787	1811 streams	0.07254	976 zeromq	0.09089	1047 comparison
0.12787	1411 ios/android	0.07239	708 referenced	0.09089	1257 standards
0.12787	1268 embedded	0.07239	658 prettyprint	0.09078	1220 reliability
0.12787	1505 messagepack	0.07239	399 false	0.09073	1301 ws-*
0.12787	1117 centralized	0.07233	1134 frameworks	0.08993	1214 queuing
0.12787	1785 solutions	0.07199	1208 overhead	0.08993	1041 cluster
0.12787	1122 choice	0.07161	225 class=lang-xml	0.0892	1019 asynchronous
0.12787	1199 cross-platform	0.07161	241 compilation	0.0892	1057 consumers
0.12787	1120 chat	0.07141	642 performance	0.08913	1190 persistent
0.12787	1390 in-memory	0.07121	1108 ef	0.08907	1072 develop
0.12787	1789 speed	0.07121	1074 concurrency	0.08737	1020 azure
0.12787	1835 syslog	0.07121	1090 delivery	0.08732	1007 akka
0.12787	1612 platform	0.07121	1310 traditional	0.08656	662 performance
0.12787	1487 lowest	0.07098	1299 technology	0.0865	1085 efficient
0.12787	1666 qpid	0.07079	710 reflection	0.08537	1110 formats
0.12787	1133 client/server	0.07079	300 debugger	0.08462	1105 fast
0.12787	1433 jetty	0.06993	932 visible	0.08293	1177 oriented
0.12787	1733 scalability	0.06993	919 var	0.08293	1199 powerful
0.12787	1324 footprint	0.06993	851 tab	0.08213	236 cmd
0.12787	1914 wf	0.06993	935 void	0.08212	1176 org/
0.12787	1041 alternitives	0.06959	1264 saml	0.08199	1179 overhead
0.12787	1472 linq	0.06903	779 servicehost	0.08196	705 queue
0.12787	1744 selection	0.06903	335 dll</code>	0.08063	1280 transactional
0.12787	1152 comet	0.06903	513 javax	0.08021	1260 subscriber

0.12787	1751 serial	0.06883	1247 real-time	0.08021	1169 netty
0.12787	1610 pitfalls	0.06869	21 </code></pre>	0.07919	200 business
0.12787	1571 nusoap	0.0683	1055 cluster	0.07861	1117 heavy
0.12787	1156 compare	0.0682	1280 speed	0.07832	1046 compared
0.12787	1731 saml	0.0682	1179 major	0.07799	1183 parsing
0.12787	1317 finagle	0.06807	933 vista	0.07794	1221 reliable
0.12787	1704 request-response	0.06712	1225 possibly	0.07764	1037 capabilities
0.12787	1333 frequent	0.06706	926 version=1	0.0772	1042 communicate
0.12787	1339 general	0.06706	944 webbrowser	0.07717	1186 patterns
0.12787	1334 front	0.06706	331 dispatcher	0.07651	681 processing
0.12787	1935 xlsx	0.06668	695 queues	0.0764	1278 topics
0.12787	1176 considerations	0.06603	1097 differences	0.07639	1132 iphone

Count	2-class dataset	4-class dataset
1	0.11825 numCOM	0.13557 distQA
2	0.1176 distQA	0.12867 numCOM
3	0.10871 numDIF	0.12533 numDIF
4	0.104 hasSourceCode	0.10559 hasSourceCode
5	0.09236 numQA	0.1051 numSUPP
6	0.09121 numSUPP	0.10327 numQA
7	0.09053 numSPED	0.0976 numVAL
8	0.08436 numVAL	0.09411 numSPED
9	0.07015 distTEC	0.0809 distTEC
10	0.0664 numPAT	0.07962 numPAT
11	0.06415 numCON	0.0777 numCON
12	0.06198 numREL	0.07409 numREL
13	0.05594 numAMT	0.07071 numREL
14	0.05394 numCHR	0.06334 numAMT
15	0.05101 numWISH	0.06221 numWISH
16	0.05072 numCOND	0.06219 numCOND
17	0.04152 numDFF	0.06168 numDFF
18	0.0411 numSOL	0.0466 numCONC
19	0.03926 numUSE	0.04632 numDIS
20	0.0392 numDIS	0.04531 numUSE
21	0.03888 numCONC	0.04151 numVS
22	0.03278 numFIT	0.04144 numSOL
23	0.02172 numVS	0.03856 numFIT
24	0.01678 numADV	0.01667 numADV
25	0.01031 numFORC	0.01649 numWORDS
26	0.01005 numSTAY	0.01608 numFORC
27	0.00863 numWORDS	0.01532 numSTAY
28	0.00786 numCOME	0.01276 numTEC
29	0.007 numTEC	0.0124 numLER
30	0.0062 numCONS	0.00723 numCOME
31	0.00592 numLER	0.00665 numASES
32	0.00462 numASES	0.00642 numCONS

Count	2-class Dataset		4-class Dataset	
1	0.08531	46 PAT TEC	0.09666	448 USE TEC
2	0.07658	51 PAT TEC TEC	0.0947	449 USE TEC
3	0.07499	127 TEC PAT	0.08923	861 USE TEC
4	0.07361	755 DIF	0.08874	293 TEC QA
5	0.06833	834 SUPP	0.08692	862 USE TEC
6	0.06724	132 TEC PAT TEC	0.08543	287 TEC TEC
7	0.062	311 QA	0.08372	897 DIF
8	0.05944	149 TEC TEC PAT	0.08359	494 DIF
9	0.05887	281 TEC VAL	0.08332	896 DIF
10	0.05785	198 TEC QA	0.08244	285 TEC TEC
11	0.05508	285 TEC SUPP	0.08058	493 DIF
12	0.05383	147 TEC TEC	0.07992	230 PAT TEC
13	0.05125	817 VAL TEC	0.07844	525 SUPP
14	0.05002	815 VAL	0.0783	231 PAT TEC
15	0.04862	134 TEC PAT TEC TEC	0.0762	781 QA
16	0.04826	753 SPED	0.07522	762 TEC TEC
17	0.04802	835 SUPP PAT	0.07455	526 SUPP
18	0.04661	257 TEC USE TEC	0.07392	761 TEC TEC
19	0.04486	291 TEC WISH	0.0731	788 QA
20	0.04318	655 USE TEC	0.07122	330 QA
21	0.04317	20 PAT	0.07096	323 QA
22	0.04247	315 QA TEC	0.06924	281 TEC PAT
23	0.04144	22 PAT PAT	0.06909	763 TEC QA
24	0.04115	271 TEC DIF	0.06648	310 TEC VAL
25	0.04094	836 SUPP TEC	0.06648	290 TEC TEC TEC
26	0.04044	288 TEC AMT	0.06646	264 TEC TEC
27	0.04037	154 TEC TEC TEC	0.06514	265 TEC TEC
28	0.03962	254 TEC USE	0.06434	718 PAT TEC
29	0.03876	757 DIF TEC	0.06324	925 SUPP
30	0.0377	168 TEC TEC QA	0.06203	917 VAL
31	0.03725	890 WISH TEC	0.06076	234 PAT
32	0.03707	REL	0.06074	746 TEC TEC
33	0.03706	446 PROG PROG PROG	0.05917	719 PAT TEC
34	0.03628	241 TEC PROG SUPP	0.05809	918 VAL
35	0.03624	659 USE TEC TEC	0.05757	517 VAL
36	0.03619	151 TEC TEC PAT TEC	0.05707	920 VAL TEC
37	0.03568	277 TEC SOL	0.05667	520 VAL TEC
38	0.03532	314 QA PAT	0.056	926 SUPP
39	0.03504	572 PROG SUPP	0.05511	759 TEC PAT
40	0.0347	857 AMT TEC	0.05494	36 COM PAT
41	0.03441	157 TEC TEC TEC TEC	0.05485	491 SPED
42	0.03403	47 PAT TEC PAT	0.05461	316 TEC WISH
43	0.03303	189 TEC TEC VAL	0.05454	314 TEC AMT
44	0.03155	644 DFF	0.05448	211 PAT COM

45	0.0314	318	QA QA	0.05395	770 TEC VAL
46	0.03088	30	PAT PAT TEC	0.05378	210 PAT COM
47	0.03041	65	PAT QA	0.05324	303 TEC USE
48	0.02989	456	PROG PROG PROG PROG	0.05301	518 VAL
49	0.02971	53	PAT TEC TEC TEC	0.05296	747 TEC TEC
50	0.02969	181	TEC TEC USE	0.05281	707 PAT COM
51	0.02961	5	REL TEC	0.0528	294 TEC QA
52	0.02858	326	QA DIF	0.0524	706 PAT COM
53	0.02852	109	PAT VAL	0.05176	1114 USE TEC TEC
54	0.02852	763	DIF USE	0.05103	1208 USE TEC TEC
55	0.02837	886	WISH PAT	0.05068	311 TEC VAL
56	0.02833	660	USE TEC TEC TEC	0.05038	722 PAT
57	0.02822	1190	QA SPED	0.0496	705 PAT
58	0.02767	768	DIS TEC	0.04904	1091 TEC DIF
59	0.02725	143	TEC PAT USE	0.04725	445 DFF
60	0.02718	190	TEC TEC SUPP	0.0468	1069 TEC PAT TEC
61	0.02695	840	SUPP QA	0.04673	209 PAT
62	0.02687	942	CONC	0.04661	312 TEC SUPP
63	0.02658	293	TEC WISH TEC	0.04617	244 PAT TEC
64	0.02658	282	TEC VAL TEC	0.04594	921 VAL TEC
65	0.02654	818	VAL TEC TEC	0.04569	317 TEC WISH
66	0.0264	1275	SUPP PAT TEC	0.04553	245 PAT TEC
67	0.02625	124	TEC REL	0.04538	1089 TEC USE TEC
68	0.02621	316	QA TEC TEC	0.04478	464 USE TEC
69	0.02618	128	TEC PAT PAT	0.04446	446 DFF
70	0.02592	188	TEC TEC SOL	0.04443	1115 USE TEC TEC
71	0.02587	319	QA PROG	0.04436	521 VAL TEC
72	0.02583	95	PAT USE	0.04422	1092 TEC DIF
73	0.02582	656	USE TEC PAT	0.04417	315 TEC AMT
74	0.02574	61	PAT TEC USE	0.04408	1037 PAT TEC TEC
75	0.02563	900	WISH QA	0.04395	153 REL
76	0.0256	286	TEC SUPP TEC	0.04333	875 USE TEC
77	0.0253	852	AMT	0.04319	528 SUPP TEC
78	0.02521	1050	PAT TEC SUPP	0.04309	40 COM TEC
79	0.02518	1155	TEC SUPP PAT	0.04284	331 QA TEC
80	0.02512	837	SUPP TEC TEC	0.04282	465 USE TEC
81	0.02473	854	AMT PAT	0.0423	1164 COND PAT
82	0.02434	155	TEC TEC TEC PAT	0.04227	492 SPED
83	0.02425	858	AMT TEC TEC	0.04226	154 REL
84	0.02406	648	USE PAT	0.04195	776 TEC WISH
85	0.02384	259	TEC USE TEC TEC	0.04195	1070 TEC PAT TEC
86	0.0232	1217	DFF TEC	0.04169	1075 TEC TEC PAT
87	0.02272	98	PAT USE TEC	0.0416	771 TEC VAL
88	0.02264	123	TEC	0.04135	220 PAT COM TEC
89	0.02263	192	TEC TEC WISH	0.04126	1189 TEC QA

90	0.02239	148 TEC TEC REL	0.04102	357 PROG SUPP
91	0.02232	182 TEC TEC USE TEC	0.04093	304 TEC USE
92	0.02226	678 USE QA	0.04092	858 DFF
93	0.02213	348 PROG PAT TEC	0.04091	774 TEC AMT
94	0.02176	816 VAL VALAT	0.04085	529 SUPP TEC
95	0.02168	152 TEC TEC PAT TEC TEC	0.04049	606 COM PAT
96	0.0214	289 TEC AMT TEC	0.04005	1184 TEC PAT TEC
97	0.02103	158 TEC TEC TEC TEC TEC	0.03991	43 COM TEC
98	0.02101	63 PAT TEC T	0.03987	1090 TEC USE TEC
99	0.02089	191 TEC TEC AMT	0.03981	1193 TEC DIF
100	0.02029	55 PAT TEC QA	0.03946	876 USE TEC
101	0.02005	1178 QA PAT TEC	0.03911	332 QA TEC
102	0.01993	676 USE TEC WISH	0.03907	930 AMT
103	0.01967	113 PAT WISH	0.03845	221 PAT COM TEC
104	0.01964	766 DIS	0.03818	859 DFF
105	0.0194	1238 SPED TEC	0.03813	284 TEC PAT
106	0.01938	4 REL PAT	0.03807	1145 SUPP PAT
107	0.01936	759 DIF QA	0.03797	1010 COM USE TEC
108	0.01935	312 QA REL	0.03791	931 AMT
109	0.01918	325 QA USE	0.03783	189 CON PAT
110	0.0188	760 DIF PROG	0.03752	278 TEC CON
111	0.01859	756 DIF PAT	0.0367	766 TEC USE
112	0.01848	574 PROG SUPP TEC	0.03667	727 PAT TEC
113	0.01822	324 QA PROG FIT	0.03658	46 COM QA
114	0.01812	892 WISH TEC TEC	0.03646	1073 TEC TEC TEC
115	0.01805	202 TEC QA DIF	0.03623	691 CON PAT
116	0.01799	789 SOL PAT	0.0359	1060 TEC SUPP
117	0.01785	1231 USE TEC AMT	0.03589	358 PROG SUPP
118	0.01783	573 PROG SUPP PAT	0.03588	1016 COND PAT
119	0.01782	791 SOL TEC	0.03586	533 AMT
120	0.01778	1075 TEC PAT TEC TEC TEC	0.0357	1291 DFF TEC
121	0.01775	985 FIT TEC	0.03569	1194 TEC DIF
122	0.01759	1049 PAT TEC P	0.03554	532 AMT
123	0.01756	1262 SOL DIF	0.03545	809 PROG SUPP
124	0.01746	1097 TEC TEC TEC SUPP	0.03537	66 COM P
125	0.01734	650 USE PAT TEC	0.03497	789 QA TEC
126	0.01723	1310 CONC QA	0.0349	777 TEC WISH
127	0.01722	62 PAT TEC USE TEC	0.03477	193 CON TEC
128	0.01684	130 TEC PAT PAT TEC	0.03466	1038 PAT TEC TEC
129	0.01683	303 TEC FIT	0.03459	466 USE TEC TEC
130	0.01679	922 WISH USE TEC	0.03451	235 PAT COM
131	0.01674	332 QA AMT	0.03446	1059 TEC SUPP
132	0.01668	421 PROG PROG	0.03446	39 COM PAT
133	0.01666	49 PAT TEC PAT TEC	0.03443	1124 DIF TEC
134	0.01655	849 SUPP AMT	0.03436	37 COM PAT COM

135	0.01644	1156 TEC SUPP TEC TEC	0.03424	837	PROG PROG
136	0.01642	156 TEC TEC TEC PAT TEC	0.03408	1172	PAT TEC TEC
137	0.01599	272 TEC DIS	0.03404	1292	DFF TEC
138	0.01584	52 PAT TEC TEC PAT	0.03403	894	SPED
139	0.0158	1051 PAT TEC AMT	0.03403	1318	COND PAT COM
140	0.01574	1126 TEC PROG SUPP TEC	0.03402	1151	WISH TEC
141	0.01574	1088 TEC TEC TEC REL	0.03396	501	DIS TEC
142	0.01569	1113 TEC TEC SUPP TEC	0.03368	1017	COND PAT COM
143	0.01564	672 USE TEC USE	0.03355	268	TEC
144	0.01561	529 PROG USE PROG	0.03352	1074	TEC TEC TEC
145	0.01559	111 PAT AMT	0.03343	750	TEC
146	0.01557	1240 SPED DIF	0.03337	313	TEC SUPP
147	0.01554	255 TEC USE PAT	0.03331	728	PAT TEC
148	0.01553	115 PAT WISH TEC	0.03314	1150	WISH TEC
149	0.0155	1265 VAL VALAT TEC	0.03313	1050	TEC TEC TEC
150	0.01545	884 WISH	0.03308	305	TEC USE TEC
151	0.01544	657 USE TEC PAT TEC	0.03303	218	PAT COM PAT
152	0.01539	110 PAT SUPP	0.03302	1106	QA QA
153	0.01536	451 PROG PROG PROG TEC	0.03286	772	TEC SUPP
154	0.01535	987 FIT TEC TEC	0.03273	571	CONC
155	0.01522	820 VAL QA	0.03272	575	CONC
156	0.01518	105 PAT DIF	0.03263	1011	COM USE TEC
157	0.01508	765 DIF WISH	0.03251	133	COND PAT
158	0.01498	787 SOL	0.03235	282	TEC PAT COM
159	0.01498	532 PROG USE PROG PROG	0.03217	775	TEC AMT
160	0.01492	860 AMT QA	0.03207	1192	TEC USE TEC
161	0.01482	144 TEC PAT USE TEC	0.03202	240	PAT CON
162	0.01481	1048 PAT TEC SOL	0.03201	540	AMT TEC
163	0.01477	481 PROG PROG USE	0.0318	1076	TEC TEC QA
164	0.01467	164 TEC TEC TEC USE	0.0317	236	PAT COM
165	0.01462	1177 QA PAT PAT	0.03165	38	COM PAT COM
166	0.01453	1278 SUPP TEC PAT	0.03163	1345	DFF TEC
167	0.01453	1266 VAL TEC VALAT	0.03162	270	TEC COM
168	0.01452	1025 PAT PAT TEC TEC	0.03162	1319	COND PAT COM
169	0.01451	1180 QA TEC PAT	0.03151	307	TEC DIS
170	0.01429	178 TEC TEC PROG SUPP	0.0315	1331	TEC SUPP
171	0.01428	196 TEC TEC FIT	0.0315	1018	COND PAT COM
172	0.01428	6 REL TEC TEC	0.03143	59	COM USE
173	0.01428	292 TEC WISH PAT	0.0314	261	TEC PAT
174	0.01424	1153 TEC VAL TEC TEC	0.03116	1096	TEC VAL TEC
175	0.01413	1188 QA PROG FIT TEC	0.03088	1185	TEC PAT
176	0.01411	327 QA DIF TEC	0.03075	1330	TEC SUPP
177	0.01401	825 VAL USE	0.03074	810	PROG SUPP
178	0.01394	23 PAT PAT PAT	0.03065	559	WISH TEC
179	0.01374	1046 PAT TEC DIF	0.03061	292	TEC TEC USE

180	0.01373	1209 PROG SUPP QA	0.03053	1202 QA TEC
181	0.01372	1098 TEC TEC TEC SUPP TEC	0.03047	269 TEC COM
182	0.01369	1274 SUPP PAT PAT	0.0304	1203 QA QA
183	0.01362	33 PAT PAT QA	0.03037	67 COM VAL
184	0.01361	207 TEC PROG PAT TEC	0.03024	1344 DFF TEC
185	0.01359	485 PROG PROG USE PROG	0.03022	902 DIS TEC
186	0.01349	459 PROG PROG PROG PROG PROG	0.03014	242 PAT PAT
187	0.01328	1089 TEC TEC TEC TEC PAT	0.03013	308 TEC SOL
188	0.01325	794 SOL QA	0.02997	1093 TEC DIS
189	0.01323	125 TEC REL TEC	0.0299	950 WISH TEC
190	0.01322	1037 PAT TEC TEC TEC TEC	0.02977	1102 QA SPED
191	0.01319	1279 SUPP TEC PAT TEC	0.02972	289 TEC TEC CON
192	0.01318	1139 TEC USE QA	0.02965	416 PROG PROG PROG
193	0.01314	666 USE TEC QA	0.02946	328 QA DIF
194	0.01313	205 TEC PROG PAT	0.02935	712 PAT COM TEC
195	0.01313	758 DIF TEC TEC	0.02923	786 QA DIF
196	0.01313	331 QA VAL	0.02923	157 REL TEC
197	0.01308	551 PROG DIF	0.0291	1198 QA SPED
198	0.013	453 PROG PROG PROG TEC PROG	0.02905	1049 TEC PAT
199	0.01298	1131 TEC DFF	0.02899	409 PROG PROG
200	0.01298	1063 PAT VAL TEC	0.02896	47 COM QA

C.3 Experiment to Search for Architectural Information in Stack Overflow

C.3.1 Architecture Searching Tasks

In order to conduct an experiment to evaluate our proposed enhanced search approach as presented in Chapter 8, practitioners solved tasks, which are attached in the following pages.

Goal of the Experiment

Compare two search engines to find relevant software architectural information on Stack-Overflow. Both search engines are classical keyword search engines. During the experiment, you will not know, which search engines are we comparing.

Experimental Procedure

Software architecture design tasks has a big scope, which involve functional and non-functional requirements, as well as constraints. You will be asked to search for architectural information on Stack Overflow to solve six different design tasks. For each task, you will use only one of two search engines. During your searching, you should record the searching queries (keywords), and the list of relevant Stack Overflow posts and their relevance to each task.

The relevance of each post (in the list of posts identified by the two search engines) to help complete the task is defined on four levels:

- **Highly Relevant** (H): The post addresses a similar problem to the task and contains useful information. The post provides an answer to the searching goal, and fulfills at least one requirement of the task.

- **Medium Relevant** (M): The post addresses another problem not similar to the task at hand, but it provides some relevant information to the task, which could be an answer to the searching goal. Nevertheless, the provided information does not consider specifically the task's requirements.

- **Low Relevance** (L): The post contains relevant information, which is only remotely relevant to solving the given task, but might help for refining the search.

- **No Relevance** (N): The post has nothing to do with the task. It has no relevant information.

When completing the tasks, you can follow several steps:

1) *Read task*: Read the task carefully to understand the requirements and the goals of the search.

2) *Log in to system*: Log in to Decision Buddy using the assigned user name and password. Please make sure that you are still logged when moving from one task to the next. The login link, user name and password are written down in each task.

3) *Access search engine*: Open the search engine given for a task. Each task has a link to the assigned search engine. Please be careful to choose the right search engine. Make sure not to use the search engine from the previous task. You will use only 1 search engine per task.

4) *Conduct search*: Start searching for the relevant posts and information that could help perform a given task. During the search:

    - Try to find as many posts as possible relevant to the task. *Submit **at least four queries** for each task. For each query, **assess only the top 10 posts***, which are returned from the search engine.

    - During assessment, **read the post carefully** to make sure that it is relevant to the problem, and do not assess the relevance just based on the title of a post. The relevance assesses the relevance between the post and the task (not the query).

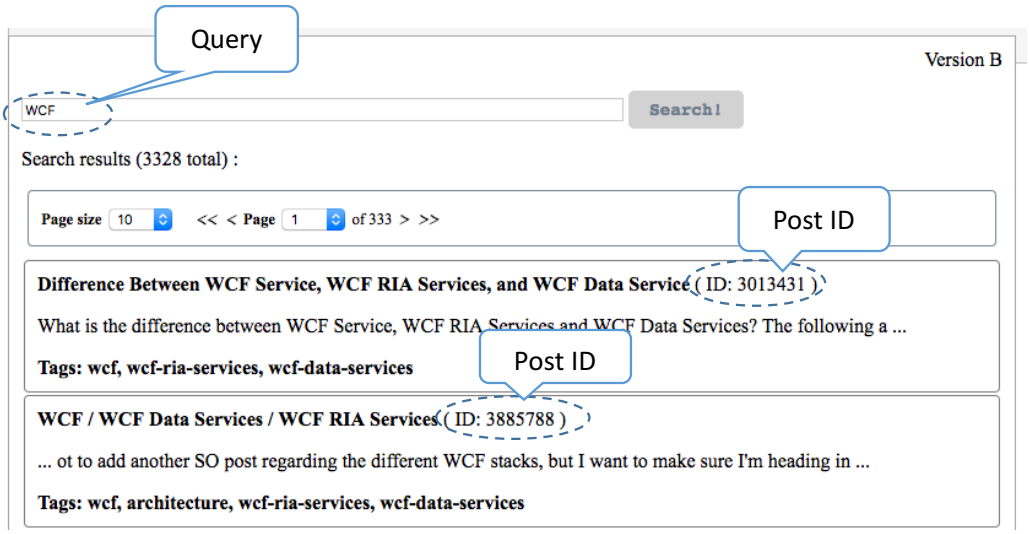
- **Do not** use any other sources of information (e.g. Google or Vendor websites) other than the assigned search engine to solve the tasks.
- Solve the tasks in their provided **sequence**.
- Try to **execute different queries** to cover all the aspects of the task (e.g. different requirements, technologies, constraints).
- **Close the browser or the tab after each task** to prevent using the wrong search engine for the following tasks.

5) *Record result*: Record your results in the attached Excel file. Record the following for each task:

- *Column "Query String (Keywords used)":* record at least five queries used for each task, even queries which didn't return any relevant posts. (See snap-shot)
- *Column "Post ID's" of relevant Stack Overflow posts* and their *relevance* level (High, Medium, Low) in column "Post Relevance". Assess only the top 10 posts per query. You do not need to record posts which have no relevance to the task. Record relevant post only one time for each task.
- *Brief explanation of why the post is relevant to the task in column "Relevance Explanation"*. You could copy parts of the post as an explanation, or write brief explanations yourself.
- Answer the question in the Excel sheet regarding the complexity of the task.

6) *Fill Survey*: Complete this questionnaire: https://goo.gl/forms/UIfekXRKHh9ImyA02

Make sure to record your information at the right excel tab. Each task has an ID in its title, as well as in the excel sheet.



Task ID: Big-Data-Stream-Evaluation

Design Scenario

An internet company provides popular content and online services to millions of web users. Besides providing information to external users, the company collects and analyzes *massive logs* of data that are generated from its infrastructure (e.g. server logs). The size of the log files is in *Tera-bytes*. To cope with the fast infrastructure growth, the company decided to develop a software application to manage the logs. A high level conceptual model has been designed, which consists of a data stream, which sends its data to a batch layer and a real time view. The data stream component dispatches data from multiple data sources in real-time. The architects of the system are discussing the possible technology choices for implementing the data stream component. Three technology families are identified as alternative architectural solutions:

1) Data collector technologies (e.g. Apache Flume, Fluentd)
2) Distributed message broker technologies (e.g. Apache Kafka, Amazon SQS, Active MQ)
3) ETL/Data Integration engines (e.g. Streamsets, Talend)

Non-functional requirements

- *Performance*: The system shall collect up to 15,000 events/second from web servers.
- *Extensibility*: The system shall support adding new data sources by just updating a configuration file.
- *Availability*: The system shall continue operating with no downtime if any single node or component fails.
- *Deployability*: The system deployment procedure shall be fully automated and support a number of environments (development, test, production).

Constraints

The system shall be composed of primary open source technologies (for cost reasons).

Search goal

The architect would like to compare the three technology families regarding their suitability to the described scenario, non-functional requirements and constraints.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user07

password: user07

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/b

Task ID: Physical-Design

Design Scenario

An IT service company builds business-to-business solutions, where heterogeneous software systems located in different locations are integrated with each other. The solution architect is working on a project, where several Customer Relationship Management (CRM) applications communicate with other software systems, which include legacy systems used in call centers, and banking systems (e.g. ATM). The CRM applications use both Microsoft and Oracle technologies. The solution architect is currently designing the integration layer, which would facilitate the communication between CRM apps and other software systems. The architect decided on Apache Camel and RabbitMQ as possible integration technologies. The selection of both technologies raises two architectural concerns:

1) Selecting a mechanism for message channeling, translation and routing.
2) Establishing a deployment topology (physical architecture).

Non-functional requirements

- *Availability*: ATM machines need high availability with no down time.
- *Performance*: The integration layer should be prepared to receive 10,000 request/sec from the CRMs.

Constraints

The company has an official agreement with Oracle for Unix servers.

Search goal

The architect would like to search for possible information on *technology features, and components design* which would help him address the aforementioned concerns.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user07

password: user07

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/b

Task ID: Conceptual-Design

Design Scenario

An online shop wants to modularize its Java web applications and expose selected components and services via APIs to external partners (e.g., marketing firms, suppliers, price comparison websites) over the internet. The lead architect of the microservices project that is in charge of this Web API design effort tries to answer the following questions:

1) What is the right service decomposition?
2) How can loose coupling and high cohesion be achieved?
3) Should communication be synchronous or asynchronous?

The architect has already read some books and articles, which (s)he found interesting but insufficient (e.g., too abstract to be actionable on the project). The architect is looking for patterns, components design and design principles for service decomposition.

Non-functional requirements

- Data consistency.
- Versioning.
- Quality of service (API security, service level agreements, performance).

Constraints

Any conceptual component that is found and any patterns should be mature, e.g., implemented in at least two different settings. Best practices, should be either applied in Java or Microsoft technologies.

Search goal

The architect would like to search for possible *architectural principles, patterns, and components design* which would help him answer the aforementioned questions.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user07

password: user07

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/b

Task ID: Middleware-Search

A stock monitoring dashboard software needs to be developed. The dashboard will be available on the internet, and could be accessed by users through mobile devices. An important aspect of the dashboard is the *real-time* change of stock values. Stock information is gathered from various sources and then transformed to be presented on the dashboard. The architect is searching for information about suitable middleware technologies, which could gather and transform the stock information.

Non-functional requirements

- *Performance*: The stock information needs to be updated in real time.
- *Scalability*: The system needs to scale to more than 100,000 users.
- *Security*: The stock information need to be securely transferred from their sources to the dashboard.

Constraints

Technologies need to be based on standard implementations with defined and published specifications.

Search goal

The architect would like to search for suitable *middleware technologies* which would fulfil the aforementioned requirements and constraints.

Find posts, which could support the architect fulfilling their request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user07

password: user07

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/c

Task ID: JSON-Search

Design Scenario

A claim management system in an insurance company is currently being modernized; a cross-platform iOS and Android mobile app has been introduced recently. This app needs to communicate with the claim management system backend (which is a JEE application hosted on an application server with IBM WebSphere) via *RESTful HTTP*; *JSON* has been decided as message exchange format. The integration architect, who is responsible for the claim management system backend and its RESTful HTTP interface now looks for *JSON parsing libraries* that come with the JEE application server (WebSphere), but also additional libraries, which could be independent from the webserver used.

Non-functional requirements

As many claims are filed and processed via this interface, the *performance of unmarshalling* (e.g., conversion from JSON to Java classes) is an important architecturally significant requirement on the server side.

Constraints

Not all open source licenses can be used: Apache 2 and Eclipse have been approved for use, GPL has been banned, all others require approval from higher-level management.

Search goal

The architect would like to search for possible *JSON parsers* for Java, which would satisfy the requirements and constraints.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user07

password: user07

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/c

Task ID: Messaging-Evaluation

Design Scenario

A help desk system is to be produced; it dispatches incoming chat messages to human agents. To save costs, a chat bot is currently being designed to replace some of these agents without compromising user satisfaction. It has been decided to implement an *update notification mechanism* in the scenario. The chat bot implementation has to be integrated with a Natural Language Processing (NLP) system and an Architectural Knowledge Base (AKB). *Asynchronous messaging* has been selected as implementation pattern for the integration channel. A *publish-subscribe* channel should be realized; this channel should be supervised and managed by using one or more systems management patterns. Three messaging technologies have been identified as candidates: 1) RabbitMQ, 2) Apache Kafka, and 3) ActiveMQ

Non-functional requirements

Guaranteed delivery of messages, high throughput and low latency are three important quality attributes. Using standard protocols and formats are required to ensure portability and interoperability.

Constraints

To avoid vendor lock-in, the chosen technology should be implemented and supported by at least three vendors. The learning curve of the technology should be in the hours-to-days range, not in the weeks-to-months range or even higher. Preferably, Java should be supported, but Python, PHP and Scala would also be acceptable.

Search goal

The architect would like to compare the three technologies regarding their suitability to the described scenario, non-functional requirements and constraints.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user07

password: user07

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/d

C.3.2 Queries to Search for Architecture Information

Searching queries for architectural information is one of our results for our experiments in Chapter 8. The queries are written by practitioners during our experiments when using search engines to find architectural information. In the following pages, we provide a list of queries for each of the design activities and design tasks.

1) Queries for Identify Architectural Solutions Design Phase

JSON-Search Task

JEE json parser
websphere json parser
websphere json library
Java json library
Java json reader
JSON java parsers
Apache 2 JSON parsers
Eclipse JSON parsers
JSON java parsers performance
JSON parsers for Java
Java REST library
Java REST framework
spring REST framework
java API management framework
java json fastest unmarsheling
java to json
RESTful json
non gpl json convert java
JSON Parser JEE open source
JSON unmarshall java REST
JSON java class REST apache eclipse
RESTful JEE application server JSON parsing
Websphere jee json parsing
json parsing java
Apache 2 json parser
Eclipse license json parser
java jackson licence
JSON parsing websphere
JSON websphere
javascript websphere
JSON parser
JSON parser jee
Jersey unmarshall
JSON pojo performance
json parsing jee websphere crossplatform
json parsing library jee websphere ios android performance
restful service json parser java performance
restful service json parser java licence eclipse apache 2
json parser java licence
json parser java websphere
json parser java websphere restful

open source licenses
json parser open source
json parser android
json parser java comparison
json java marshalling
json parser performance
performance unmarshal
JSON JEE conversion
JSON parser
JSON parser Java
RESTful HTTP JSON to Java object
high performance
jave
apache 2 license
Apache JSON parsers for Java
Jason Java Parser
Jason Parser
fast unmarshalling java
json performance
High performance json libs java
performance unmarshaling flexjson
Jersey vs flexjson

Middleware-Search Task

Real time middleware framework
Message queue framework benchmark
large scale realtime dashboard
large scale data transport layer
Middleware technologies for transferring data
High performance Middleware technologies for transferring data
Middleware technologies Scalability
Middleware technologies security
middleware service design
middleware real-time solution
scalable middleware real-time solution
secure design for realtime middleware solution
data crawling
transform data multi source
middleware scalability
middleware comparison
performance middleware
open source middleware low latency
scalable middleware 100000 users
middleware security financial

middleware data processing realtime
Stock market real time retrieval
end to end encrypting api calls
dashboard high performance
scalable ui design
middleware stock analysis provider
middleware real-time
middleware stock exchange
middleware stock market
mqtt performance
rabbitmq mobile
amqp mobile
rabbitmq Scalability
real time update middleware scalable
real time update middleware security secure
middleware security secure
middleware security secure messaging
middleware redis realtime update
middleware zeromq realtime update secure scalable
real time application
security sending data
real time scalability
real time performance
Dashboard
middleware monitoring
middleware security
middleware performance
middleware dashboard
data integration transformation
real time
scalable
secure
middleware
large scale real time middleware
enterprise realtime messaging
enterprise realtime dashboard system
large scale realtime dashboard system
real-time middleware dashboard
real-time stock dashboard
stock middleware
ZeroMQ performamce
scalable stock middleware
ZeroMQ performance

2) Queries for Select Architectural Solutions Design Phase

Messaging-Evaluation Task

Latency and reliability of RabbitMQ
Latency and reliability of Apache Kafka
Latency and reliability of ActiveMQ
RabitMQ vs ActiceMQ
RabitMQ Java Integration
Java message queue technologies
RabbitMQ throughput
Apache Kafka throughput
ActiveMQ throughput
Ensure guaranteed delivery of messages using RabbitMQ
RabbitMQ vs Apache Kafka vs ActiveMQ
publish subscribe reliable messaging system
reliable messaging system
reliable high performance messaging system
realtime messaging system
RabbitMQ vs ActiveMQ throughput
Apache Kafka performance
java python jms
scala activemq
RabbitMQ vs Kafka vs ActiveMQ
KAFKA supported libraries
using RABBITMQ with java
ACTIVEMQ supported languages
ActiveMQ vs kafka vs RabbitMQ
Kafka maintainability
Kafka vs amazon kinesis
does ActiveQ support python
RabbitMQ latency
RabbitMQ Apache Kafka ActiveMQ comparison
kafka publish subscribe
Guaranteed delivery message
activemq vendor support
rabbitmq vendor support
rabbitmq vendor
rabbitmq kafka activemq
rabbitmq kafka activemq delivery messages high throughput low latency
chat application publish subscribe asynchronous
chat application publish subscribe asynchronous java portability interoperability
chat application publish subscribe asynchronous java activemq kafka rabbitmq high throughput
activemq kafka rabbitmq vendor
messaging brokers
messaging brokers java

messaging brokers python
low latency messaging
RabbitMQ versus ActiveMQ
Kafka versus ActiveMQ
Kafka versus RabbitMQ
best messaging technology
RabbitMQ vs Apache Kafka vs ActiveMQ
Guaranteed delivery
high throughput
low latency
RabbitMQ on apatche
RabbitMQ Apache Kafka ActiveMQ
high throughput and low latency messaging
easy to learn messaging high throughput and low latency
RabbitMQ vs. Apache Kafka vs. ActiveMQ
RabbitMQ using Java PHP Python
Apache Kafka
ActiveMQ
ActiveMQ JAVA
RabbitMQ JAVA
messaging RabbitMQ
Compare RabbitMQ , Apache Kafka , ActiveMQ
RabbitMQ , Apache Kafka , ActiveMQ vendors
RabbitMQ vendors
RabbitMQ activemq kafka
kafka support
message support many vendors
kafka
learning kafka

Big-Data-Stream-Evaluation Task

Data Collector vs Message Queue brokers
collecting high amount of log events
performance benchmark of FluentID
performance benchmark of Apache Flume
performance benchmark of Streamsets
Data collector technologies vs Distributed message broker technologies
Apache Flume VS Apache Kafka from performance point of view
Apache Flume VS Apache Flume from Availability point of view
Apache Flume VS Apache Flume from deployability point of view
data streaming tools
data collector streaming tools
distributed message broker streaming tools
data streaming ETL Data Integration engines

high availability data streaming solution
kafka vs flume
kafka vs streamsets
Apache data streaming solutions
ETL analyze big logs
best data collector framework
Apache Flume vs Fluentd
Talend best practices
Open source ETL engine
Tera logs byte analytics
analyze application log
flume Fluentd performance
flume Fluentd
Apache Kafka, Amazon SQS, Active MQ
Streamsets Talend
high end open source kafka flume sqs talend
Amazon SQS vs Fluentd
Amazon SQS pricing
Streamsets vs talend
Apache Flume
Flume performance
kakfa performance
data collector
kafka talend
talend message broker
streamsets message broker
log data stream
message broker etl
big data massive log data multiple data sources data streams
big data massive log data multiple data sources data streams apache flume
big data log data analysis multiple data sources data extensible performance availability deployability
apache flume fluentd
data integration engine for big data
distributed message broker for big data
apache flume vs apache kafka vs streamsets
big data stream logging analysis best technology server logs
availability Flume
scalability messaging broker
ETL
scalability flume
stream big data technologies
data collector versus distributed message broker versus ETL data integration
flume kafka streamsets
flume kafka streamsets big data
streaming big data

open source engines
real-time
extensible
high-performance
Data collector
Distributed message broker
ETL Data Integration
open source data collector high scale
open source data collector high performance
open source data collector products
high availability salable automation deploy open source data collector products
Apache Flume vs. Apache Kafka vs. Talend
Fluentd vs. Amazon SQS vs. Streamsets
Fluentd vs. Amazon SQS vs. Streamsets deployment
Fluentd performance
Kafka performance
Deploying Kafka
Amazon SQS performance
Big data stream
Distributed message broker
Data Integration engines
Streamsets and Talend
Apache Flume vs Amazon SQS Streamsets

3) Queries for Instantiate Architectural Elements Design Phase

Physical-Design Task

RabbitMq .NET integration
Apache Camel .NEt integration
Apache Camel vs RabbitMQ integration performance
Apache Camel performance benchemark
RabbitMQ unix scalability
Apache Camel performance for unix servers
RabbitMQ performance for unix servers
Apache Camel and oracle for unix servers
RabbitMQ and oracle for unix servers
Apache Camel vs RabbitMQ
RabbitMQ physical architecture
Architecture guidance for a Rabbitmq and Apache camel
Architecture guidance for a Rabbitmq
Architecture guidance for apache camel
high availability messaging engines
rabbitmq
RabbitMQ High Availablility

Apache Camel Microsoft
Apache Camel Unix
Apache Camel Rest SOAP
ActiveMQ with .NET
apache camel deployment unix sun
rabbitMQ clustering high availability
high performance deployment of messaging broker camel rabbitMq
message channeling, translation and routing
Highly available integration technologies
Apache camel performance
Biztalk vs camel
apache camel deployment
camel messaging channel
rabbitmq message channel
rabbitmq performance
rabbitmq oracle
rabbitmq deployment
rabbit mq deployment
apache camel translations
apache camel rabbitmq
integration layer communication heterogeneous software systems apache camel rabbitmq message channelin
integrating legacy systems apache camel message channel
apache camel vs rabbitmq integration legacy availability
apache camel vs rabbitmq integration legacy availability features components
apache camel vs rabbitmq constraint oracle unix
Camel availability
camel performance
camel rabbitMQ
Camel Fault tolerance
camel activeMQ
deployment topology apache camel rabbitMQ
message channel rabbitMQ
message channel apache camel
routing apache camel rabbitMQ
apache camel versus rabbitMQ
performance
availability
linux support
Apache Camel and RabbitMQ availablity
Apache Camel and RabbitMQ and availability and performance
best integration technology availability and performance
oracle unix best integration technology availability and performance
CRM Integration technology
CRM and RabbitMQ
CRM integration requirement

CRM integration and deployment topology
deployment topology
message channeling rabbitmq
physical architecture
RabbitMQ Camel

Conceptual-Design Task

services decomposition patterns
web apis design pattern
web apis versioning
extensible web service pattern
architectural principles for micro services
architectural patterns for micro services
loose-coupling and high cohesion for micro services
synchronous vs asynchronous calls for micro services
microservices decomposition best practice
microservices design best practice
micro-services architecture
micro services framework
micro-services framework
micro-services framework webapi
REST API best practice
API security
micro-services communication patterns
microservices security practices
microservices communication layer
microservices sync async
Microservices design pattern
Micro services design
Design Patterns service decomposition
loose coupling and high cohesion
synchronous vs asynchronous micro service
API design performance security
java microservice decomposition
microservice architecture patterns
database service concurency
api authentication java
spring cor service
service decomposition
modularization pattern
microservice pattern
decompose pattern
loose coupling high cohesion
loose coupling micro service

messaging micro service
 micro service architecture
component design
architecture rest service
decomposition
service cutting
pattern service decomposition
pattern service decomposition web api microservice design loose coupling
service api high cohesion modules
SOA API design patterns
SOA patterns service security
architecture coupling
Asynchronous vs synchronous
Asynchronous advantages
expose components
SOA java
pattern decomposition microservices
good API design
webservice API design
web API design asynchronous communication
data consistency web api microservice
 micro-services
 design patterns principles
 patterns and components design
 design patterns architecture
microservice architecture
architecture design patterns
web api versioning
data consistency synchronous asynchronous
microservices ensure data consistency
microservices versioning
microservices security
microservices component communication
SOA Design Pattern
loose coupling in service decomposition
service decomposition performance and security
Ecommerce microservice
microservice design
synchronous vs asynchronous service communication
decomposing services and data conssistency

C.3.3 Relevant Posts for Searching Tasks

Relevant posts to architecture design tasks is one of our results for the experiments in Chapter 8. In the next following pages, we list the posts identified by each practitioner for being relevant to the tasks. Each Stack Overflow post is identified by an ID. You may access the full post by concatenating the URL `stackoverflow.com/questions/` with the post ID. Moreover, the relevance of each post to the design tasks is specified by each practitioner as defined in Chapter 8.

Practitioner	Task	Post ID	Post Relevance
1	Middleware-Search	1112279	3
1	Middleware-Search	2388539	2
1	Middleware-Search	1919472	3
1	JSON-Search	13993862	3
1	JSON-Search	2253750	1
1	JSON-Search	3471013	1
1	Messaging-Evaluation	2124221	3
1	Messaging-Evaluation	6939110	3
1	Messaging-Evaluation	1919472	2
1	Messaging-Evaluation	27666943	2
1	Messaging-Evaluation	12559570	3
1	Messaging-Evaluation	6664445	2
1	Messaging-Evaluation	7044157	3
1	Messaging-Evaluation	2705043	1
1	Messaging-Evaluation	2166590	3
1	Big-Data-Stream-Evaluation	2166590	2
1	Big-Data-Stream-Evaluation	1840684	3
1	Big-Data-Stream-Evaluation	7921324	3
1	Big-Data-Stream-Evaluation	2124221	3
1	Big-Data-Stream-Evaluation	7382655	1
1	Big-Data-Stream-Evaluation	17708489	3
1	Big-Data-Stream-Evaluation	296650	2
1	Big-Data-Stream-Evaluation	12559570	3
1	Big-Data-Stream-Evaluation	3202521	2
1	Physical-Design	2225761	3
1	Physical-Design	10407760	3
1	Physical-Design	10030227	3
1	Physical-Design	4971437	3
1	Conceptual-Design	9587393	2
1	Conceptual-Design	1670332	2
1	Conceptual-Design	445093	3
1	Conceptual-Design	11386370	3
1	Conceptual-Design	39585	3
1	Conceptual-Design	409338	3
1	Conceptual-Design	36999	3
1	Conceptual-Design	9672334	3
1	Conceptual-Design	10592078	3
1	Conceptual-Design	18936264	3
1	Conceptual-Design	8885514	3
1	Conceptual-Design	8703854	3
1	Conceptual-Design	447518	3
1	Conceptual-Design	3051326	3
2	Middleware-Search	2388539	3
2	Middleware-Search	8848058	3

2	Middleware-Search	6664445	3
2	Middleware-Search	21808529	3
2	Middleware-Search	296650	3
2	Middleware-Search	11840630	3
2	Middleware-Search	9296466	3
2	Middleware-Search	6939110	3
2	Middleware-Search	43823	3
2	Middleware-Search	567478	2
2	Middleware-Search	2748115	1
2	Middleware-Search	1920959	3
2	Middleware-Search	1558207	3
2	JSON-Search	3471013	1
2	JSON-Search	57689	2
2	JSON-Search	803515	2
2	JSON-Search	2782076	2
2	JSON-Search	7433720	1
2	JSON-Search	1237649	2
2	JSON-Search	4163066	2
2	JSON-Search	5145129	2
2	JSON-Search	6939110	1
2	Messaging-Evaluation	10030227	3
2	Messaging-Evaluation	9685235	2
2	Messaging-Evaluation	15150017	3
2	Messaging-Evaluation	9569851	3
2	Messaging-Evaluation	6939110	3
2	Messaging-Evaluation	10407760	3
2	Messaging-Evaluation	2124221	3
2	Messaging-Evaluation	18531072	3
2	Messaging-Evaluation	12687368	2
2	Messaging-Evaluation	21808529	1
2	Messaging-Evaluation	20520492	2
2	Messaging-Evaluation	16449126	1
2	Messaging-Evaluation	12130481	3
2	Messaging-Evaluation	27666943	2
2	Messaging-Evaluation	6664445	2
2	Messaging-Evaluation	2225761	3
2	Messaging-Evaluation	22977299	2
2	Messaging-Evaluation	731233	1
2	Messaging-Evaluation	14401632	3
2	Messaging-Evaluation	2279417	1
2	Messaging-Evaluation	21363302	2
2	Messaging-Evaluation	11926077	3
2	Messaging-Evaluation	3760100	3
2	Big-Data-Stream-Evaluation	3963362	2
2	Big-Data-Stream-Evaluation	773503	2

2	Big-Data-Stream-Evaluation	21808529	3
2	Big-Data-Stream-Evaluation	13483809	1
2	Big-Data-Stream-Evaluation	240471	2
2	Big-Data-Stream-Evaluation	12634965	3
2	Big-Data-Stream-Evaluation	12559570	3
2	Big-Data-Stream-Evaluation	10445621	1
2	Big-Data-Stream-Evaluation	27666943	3
2	Big-Data-Stream-Evaluation	6930236	2
2	Big-Data-Stream-Evaluation	3792519	2
2	Big-Data-Stream-Evaluation	10330998	1
2	Big-Data-Stream-Evaluation	11139400	3
2	Physical-Design	9723682	1
2	Physical-Design	3792519	1
2	Physical-Design	8845186	1
2	Physical-Design	6939110	2
2	Physical-Design	7921324	2
2	Physical-Design	10407760	2
2	Physical-Design	2124221	2
2	Physical-Design	10030227	2
2	Conceptual-Design	2897513	2
2	Conceptual-Design	2567254	1
2	Conceptual-Design	4481311	1
2	Conceptual-Design	2592609	2
2	Conceptual-Design	2681318	1
2	Conceptual-Design	2508361	1
2	Conceptual-Design	945123	1
2	Conceptual-Design	20773875	2
2	Conceptual-Design	17401679	3
2	Conceptual-Design	447518	1
2	Conceptual-Design	10641999	1
2	Conceptual-Design	2503071	3
2	Conceptual-Design	255499	2
2	Conceptual-Design	11755146	2
2	Conceptual-Design	514306	1
3	Middleware-Search	2388539	2
3	Middleware-Search	1112279	2
3	Middleware-Search	9735578	2
3	Middleware-Search	2936598	3
3	Middleware-Search	24176969	1
3	Middleware-Search	8845186	2
3	JSON-Search	5015479	3
3	JSON-Search	3471013	3
3	JSON-Search	57689	2
3	JSON-Search	19758215	3
3	JSON-Search	4003102	2

3	Messaging-Evaluation	27666943	2
3	Messaging-Evaluation	32851	2
3	Messaging-Evaluation	2669573	2
3	Messaging-Evaluation	2124221	3
3	Messaging-Evaluation	4559883	1
3	Messaging-Evaluation	14241947	2
3	Big-Data-Stream-Evaluation	20501430	1
3	Big-Data-Stream-Evaluation	27666943	3
3	Big-Data-Stream-Evaluation	21808529	2
3	Big-Data-Stream-Evaluation	12130481	2
3	Physical-Design	3792519	1
3	Physical-Design	7877461	1
3	Physical-Design	4406656	1
3	Physical-Design	2124221	3
3	Physical-Design	10745084	2
3	Conceptual-Design	10173933	1
3	Conceptual-Design	10592078	1
3	Conceptual-Design	24981806	1
4	Middleware-Search	1112279	1
4	Middleware-Search	2388539	2
4	Middleware-Search	13483809	2
4	Middleware-Search	10334306	3
4	Middleware-Search	10055290	3
4	Middleware-Search	17806977	2
4	JSON-Search	3471013	3
4	JSON-Search	2253750	1
4	JSON-Search	9314735	2
4	JSON-Search	13594945	1
4	JSON-Search	9292424	2
4	JSON-Search	15609306	1
4	JSON-Search	17049684	2
4	Messaging-Evaluation	731233	3
4	Messaging-Evaluation	21808529	3
4	Messaging-Evaluation	12130481	3
4	Messaging-Evaluation	21293937	3
4	Messaging-Evaluation	20520492	3
4	Messaging-Evaluation	24863598	3
4	Messaging-Evaluation	6551718	3
4	Messaging-Evaluation	6551718	3
4	Messaging-Evaluation	7044157	1
4	Big-Data-Stream-Evaluation	2388539	1
4	Big-Data-Stream-Evaluation	12559570	2
4	Big-Data-Stream-Evaluation	27666943	1
4	Big-Data-Stream-Evaluation	26623673	2
4	Physical-Design	6577218	3

4	Physical-Design	3413424	3
4	Physical-Design	10030227	2
4	Physical-Design	3758576	1
4	Physical-Design	10407760	2
4	Physical-Design	2225761	2
4	Physical-Design	1839576	1
4	Physical-Design	587899	1
4	Conceptual-Design	162376	2
4	Conceptual-Design	15054777	1
4	Conceptual-Design	11755146	1
4	Conceptual-Design	2567254	3
5	Middleware-Search	2363157	2
5	Middleware-Search	6233364	2
5	Middleware-Search	11158543	1
5	Middleware-Search	1919472	2
5	Middleware-Search	1112279	3
5	Middleware-Search	61437	3
5	Middleware-Search	5757886	3
5	Middleware-Search	1621532	1
5	Middleware-Search	9285953	2
5	Middleware-Search	8848058	2
5	Middleware-Search	12966460	1
5	Middleware-Search	4270883	2
5	Middleware-Search	13483809	3
5	Middleware-Search	7919763	1
5	Middleware-Search	21808529	3
5	Middleware-Search	2388539	2
5	Middleware-Search	25550819	1
5	Middleware-Search	10055290	2
5	Middleware-Search	9043802	1
5	JSON-Search	13993862	1
5	JSON-Search	21367158	2
5	JSON-Search	5820028	2
5	JSON-Search	5015479	1
5	JSON-Search	13953325	2
5	JSON-Search	11465653	2
5	JSON-Search	10323957	2
5	JSON-Search	12048804	2
5	JSON-Search	8805802	2
5	JSON-Search	15565862	2
5	JSON-Search	13594945	1
5	Messaging-Evaluation	27666943	3
5	Messaging-Evaluation	7044157	3
5	Messaging-Evaluation	12559570	3
5	Messaging-Evaluation	6664445	2

5	Messaging-Evaluation	12130481	2
5	Messaging-Evaluation	32851	2
5	Messaging-Evaluation	6636213	2
5	Messaging-Evaluation	17708489	2
5	Messaging-Evaluation	507391	2
5	Messaging-Evaluation	8261654	1
5	Messaging-Evaluation	3034054	2
5	Messaging-Evaluation	6939110	3
5	Messaging-Evaluation	15150133	1
5	Messaging-Evaluation	7382655	1
5	Messaging-Evaluation	3280576	1
5	Messaging-Evaluation	2124221	2
5	Messaging-Evaluation	5132648	1
5	Messaging-Evaluation	22989833	2
5	Messaging-Evaluation	21098502	2
5	Messaging-Evaluation	12805377	1
5	Messaging-Evaluation	2705043	1
5	Messaging-Evaluation	4559883	2
5	Messaging-Evaluation	9060402	1
5	Big-Data-Stream-Evaluation	12559570	3
5	Big-Data-Stream-Evaluation	1809296	1
5	Big-Data-Stream-Evaluation	13681213	2
5	Big-Data-Stream-Evaluation	26623673	2
5	Big-Data-Stream-Evaluation	27666943	3
5	Big-Data-Stream-Evaluation	12130481	2
5	Big-Data-Stream-Evaluation	10593739	1
5	Big-Data-Stream-Evaluation	17708489	2
5	Big-Data-Stream-Evaluation	507391	2
5	Big-Data-Stream-Evaluation	3151966	1
5	Big-Data-Stream-Evaluation	7448677	2
5	Big-Data-Stream-Evaluation	2429844	2
5	Big-Data-Stream-Evaluation	8182543	2
5	Big-Data-Stream-Evaluation	4604823	2
5	Big-Data-Stream-Evaluation	11685569	2
5	Big-Data-Stream-Evaluation	5194446	2
5	Big-Data-Stream-Evaluation	2124221	1
5	Physical-Design	6061813	1
5	Physical-Design	2225761	3
5	Physical-Design	3413424	3
5	Physical-Design	10030227	3
5	Physical-Design	4971437	2
5	Physical-Design	11248510	2
5	Physical-Design	10407760	2
5	Physical-Design	6169658	1
5	Physical-Design	21098502	2

5	Physical-Design	10745084	1
5	Physical-Design	18531072	1
5	Physical-Design	3465675	1
5	Physical-Design	8261654	1
5	Physical-Design	3758576	1
5	Physical-Design	12147614	3
5	Conceptual-Design	10641999	1
5	Conceptual-Design	4368482	1
5	Conceptual-Design	8580002	1
5	Conceptual-Design	2503071	3
5	Conceptual-Design	13879379	2
5	Conceptual-Design	309374	2
5	Conceptual-Design	753273	2
5	Conceptual-Design	20501430	1
5	Conceptual-Design	22638828	3
5	Conceptual-Design	8065019	2
5	Conceptual-Design	9623482	3
5	Conceptual-Design	6147288	1
5	Conceptual-Design	2554999	2
5	Conceptual-Design	5979252	2
5	Conceptual-Design	1890679	2
5	Conceptual-Design	580514	1
5	Conceptual-Design	5145129	2
5	Conceptual-Design	4163066	1
5	Conceptual-Design	24981806	2
5	Conceptual-Design	5824171	1
5	Conceptual-Design	10172506	2
5	Conceptual-Design	11386370	2
6	Middleware-Search	417453	3
6	Middleware-Search	2912051	2
6	Middleware-Search	3164821	3
6	Middleware-Search	12099446	2
6	Middleware-Search	2936598	2
6	Middleware-Search	3198781	3
6	Middleware-Search	11378046	3
6	Middleware-Search	11840630	2
6	Middleware-Search	296650	3
6	Middleware-Search	4163066	2
6	Middleware-Search	13592894	3
6	Middleware-Search	1670332	3
6	Middleware-Search	2936598	3
6	Middleware-Search	4801545	2
6	Middleware-Search	4648280	3
6	Middleware-Search	23280761	2
6	Middleware-Search	11829551	2

6	Middleware-Search	2630492	2
6	JSON-Search	5015479	2
6	JSON-Search	3471013	3
6	JSON-Search	6518401	2
6	JSON-Search	57689	3
6	JSON-Search	3213398	2
6	Messaging-Evaluation	27666943	3
6	Messaging-Evaluation	731233	3
6	Messaging-Evaluation	7044157	3
6	Messaging-Evaluation	12559570	3
6	Messaging-Evaluation	6664445	2
6	Messaging-Evaluation	21808529	2
6	Messaging-Evaluation	20520492	2
6	Messaging-Evaluation	26623673	2
6	Messaging-Evaluation	3280676	2
6	Messaging-Evaluation	6551718	3
6	Messaging-Evaluation	10030227	3
6	Messaging-Evaluation	27830955	2
6	Messaging-Evaluation	6362829	3
6	Messaging-Evaluation	2124221	3
6	Messaging-Evaluation	9569851	3
6	Messaging-Evaluation	20520492	2
6	Messaging-Evaluation	6481760	3
6	Messaging-Evaluation	6939110	3
6	Big-Data-Stream-Evaluation	12559570	3
6	Big-Data-Stream-Evaluation	27666943	2
6	Big-Data-Stream-Evaluation	21808529	3
6	Big-Data-Stream-Evaluation	18202986	3
6	Big-Data-Stream-Evaluation	2336438	2
6	Big-Data-Stream-Evaluation	8182543	2
6	Big-Data-Stream-Evaluation	7448677	2
6	Big-Data-Stream-Evaluation	2429844	3
6	Physical-Design	292860	3
6	Physical-Design	3034054	3
6	Physical-Design	376127	2
6	Physical-Design	12008	3
6	Physical-Design	15204051	3
6	Physical-Design	8845186	3
6	Physical-Design	3921436	3
6	Physical-Design	10330998	2
6	Physical-Design	3792519	2
6	Physical-Design	4098813	2
6	Physical-Design	9723682	2
6	Physical-Design	378088	2
6	Physical-Design	12471698	2

6	Physical-Design	3355082	2
6	Physical-Design	5660541	2
6	Physical-Design	10330998	2
6	Conceptual-Design	514306	2
6	Conceptual-Design	2392650	2
6	Conceptual-Design	10055290	3
6	Conceptual-Design	1365357	3
6	Conceptual-Design	722675	3
6	Conceptual-Design	3836900	3
6	Conceptual-Design	5044585	2
6	Conceptual-Design	2613348	2
6	Conceptual-Design	969964	2
6	Conceptual-Design	226953	2
6	Conceptual-Design	2116799	2
6	Conceptual-Design	8148702	3
6	Conceptual-Design	773503	2
6	Conceptual-Design	162376	2
6	Conceptual-Design	1296460	3
6	Conceptual-Design	2737685	2
6	Conceptual-Design	6302341	2
6	Conceptual-Design	30156407	2
6	Conceptual-Design	10055290	2
6	Conceptual-Design	268129	2
6	Conceptual-Design	8926506	2
6	Conceptual-Design	19560479	2
6	Conceptual-Design	4906369	2
6	Conceptual-Design	331419	2
6	Conceptual-Design	5349428	2
7	Middleware-Search	1112279	1
7	Middleware-Search	2912051	1
7	Middleware-Search	24176969	3
7	Middleware-Search	417453	1
7	Middleware-Search	5132648	2
7	Middleware-Search	7129821	1
7	Middleware-Search	10057189	2
7	Middleware-Search	12147614	3
7	Middleware-Search	22989833	2
7	JSON-Search	57689	2
7	JSON-Search	24684958	3
7	JSON-Search	30156407	2
7	Messaging-Evaluation	27666943	2
7	Messaging-Evaluation	7044157	2
7	Messaging-Evaluation	12559570	2
7	Messaging-Evaluation	8261654	3
7	Messaging-Evaluation	3015178	3

7	Messaging-Evaluation	12130481	1
7	Messaging-Evaluation	12805377	2
7	Big-Data-Stream-Evaluation	12559570	2
7	Big-Data-Stream-Evaluation	21808529	2
7	Physical-Design	14195898	1
7	Physical-Design	10953237	2
7	Physical-Design	5681118	2
7	Physical-Design	22989833	2
7	Physical-Design	5950322	1
7	Physical-Design	8548983	1
7	Conceptual-Design	447518	1
7	Conceptual-Design	2503071	3
8	Middleware-Search	2912051	3
8	Middleware-Search	2388539	1
8	Middleware-Search	12733985	3
8	Middleware-Search	16532449	2
8	Middleware-Search	7506118	3
8	Middleware-Search	5391435	3
8	JSON-Search	13295412	1
8	JSON-Search	17105322	2
8	JSON-Search	8805802	1
8	JSON-Search	9389842	1
8	JSON-Search	13594945	2
8	JSON-Search	57689	2
8	JSON-Search	3471013	2
8	Messaging-Evaluation	27666943	2
8	Messaging-Evaluation	731233	2
8	Messaging-Evaluation	20520492	2
8	Messaging-Evaluation	2124221	2
8	Messaging-Evaluation	1919472	2
8	Messaging-Evaluation	7044157	2
8	Big-Data-Stream-Evaluation	1671302	1
8	Big-Data-Stream-Evaluation	13483809	2
8	Big-Data-Stream-Evaluation	21808529	1
8	Big-Data-Stream-Evaluation	12559570	1
8	Big-Data-Stream-Evaluation	1809296	2
8	Physical-Design	10407760	1
8	Physical-Design	6061813	3
8	Physical-Design	10030227	2
8	Conceptual-Design	2503071	3
8	Conceptual-Design	39585	2
8	Conceptual-Design	447518	3
8	Conceptual-Design	1296460	3
9	Middleware-Search	24176969	1
9	Middleware-Search	4823191	1

9	Middleware-Search	7193014	3
9	Middleware-Search	2936598	2
9	Middleware-Search	3164821	2
9	Middleware-Search	12733985	3
9	Middleware-Search	11021554	2
9	Middleware-Search	6169658	3
9	Middleware-Search	12966460	3
9	Middleware-Search	2912051	3
9	Middleware-Search	3357457	3
9	JSON-Search	57665	1
9	JSON-Search	1763099	1
9	JSON-Search	1237649	1
9	JSON-Search	15056878	1
9	JSON-Search	684515	1
9	JSON-Search	2782076	2
9	JSON-Search	5687650	1
9	JSON-Search	7332029	2
9	JSON-Search	2992258	2
9	JSON-Search	57689	3
9	JSON-Search	282644	3
9	JSON-Search	4163066	1
9	Messaging-Evaluation	8261654	2
9	Messaging-Evaluation	507391	3
9	Messaging-Evaluation	17708489	3
9	Messaging-Evaluation	6636213	3
9	Messaging-Evaluation	32851	3
9	Messaging-Evaluation	12130481	3
9	Messaging-Evaluation	6664445	1
9	Messaging-Evaluation	12559570	2
9	Messaging-Evaluation	7044157	3
9	Messaging-Evaluation	27666943	3
9	Messaging-Evaluation	15150133	2
9	Messaging-Evaluation	2124221	3
9	Messaging-Evaluation	1919472	3
9	Messaging-Evaluation	12805377	3
9	Messaging-Evaluation	22989833	3
9	Messaging-Evaluation	6939110	3
9	Big-Data-Stream-Evaluation	6939110	3
9	Big-Data-Stream-Evaluation	12559570	3
9	Big-Data-Stream-Evaluation	7382655	3
9	Big-Data-Stream-Evaluation	17708489	3
9	Big-Data-Stream-Evaluation	12130481	3
9	Big-Data-Stream-Evaluation	24863598	3
9	Physical-Design	3792519	3
9	Physical-Design	19487835	3

9	Physical-Design	9413045	3
9	Physical-Design	14501119	3
9	Physical-Design	7044157	2
9	Physical-Design	2279417	3
9	Physical-Design	14401632	3
9	Physical-Design	2225761	3
9	Physical-Design	6065062	3
9	Conceptual-Design	3488395	3
9	Conceptual-Design	5971252	3
9	Conceptual-Design	13953733	1
9	Conceptual-Design	698998	2
9	Conceptual-Design	3836900	3
9	Conceptual-Design	969964	3
9	Conceptual-Design	2503071	3
9	Conceptual-Design	8639625	1
9	Conceptual-Design	8065019	3
9	Conceptual-Design	2096734	3
9	Conceptual-Design	447518	2
9	Conceptual-Design	10055290	3
9	Conceptual-Design	8148702	3
9	Conceptual-Design	1296460	3
10	Middleware-Search	9378293	2
10	Middleware-Search	1112279	2
10	JSON-Search	3471013	2
10	JSON-Search	15565862	2
10	Messaging-Evaluation	7044157	3
10	Messaging-Evaluation	731233	2
10	Messaging-Evaluation	27666943	2
10	Messaging-Evaluation	12130481	1
10	Messaging-Evaluation	20520492	1
10	Big-Data-Stream-Evaluation	12559570	2
10	Big-Data-Stream-Evaluation	30098076	1
10	Physical-Design	3758576	1
10	Physical-Design	4444208	1
10	Physical-Design	18531072	2
10	Physical-Design	10407760	2
10	Physical-Design	10745084	1
10	Physical-Design	10030227	1
10	Conceptual-Design	39585	1
10	Conceptual-Design	4322819	1
10	Conceptual-Design	10641999	1
10	Conceptual-Design	2567254	1
11	Middleware-Search	1074800	1
11	Middleware-Search	2388539	2
11	Middleware-Search	13483809	3

11	Middleware-Search	10055290	1
11	Middleware-Search	1112279	1
11	Middleware-Search	374464	1
11	Middleware-Search	21808529	3
11	Middleware-Search	12008	1
11	Middleware-Search	3034054	1
11	Middleware-Search	8845186	1
11	JSON-Search	2782076	1
11	JSON-Search	8964424	1
11	JSON-Search	296650	2
11	JSON-Search	3792519	1
11	JSON-Search	6939110	1
11	JSON-Search	11840630	1
11	JSON-Search	21808529	1
11	JSON-Search	330174	1
11	JSON-Search	10334306	1
11	JSON-Search	14580337	1
11	Messaging-Evaluation	27666943	2
11	Messaging-Evaluation	20520492	1
11	Messaging-Evaluation	731233	3
11	Messaging-Evaluation	21808529	1
11	Messaging-Evaluation	6664445	1
11	Messaging-Evaluation	2124221	1
11	Messaging-Evaluation	10030227	1
11	Messaging-Evaluation	16449126	1
11	Messaging-Evaluation	9569851	1
11	Messaging-Evaluation	1919472	1
11	Big-Data-Stream-Evaluation	773503	2
11	Big-Data-Stream-Evaluation	2486721	1
11	Big-Data-Stream-Evaluation	12634965	1
11	Big-Data-Stream-Evaluation	7921324	1
11	Big-Data-Stream-Evaluation	2388539	1
11	Big-Data-Stream-Evaluation	2124221	1
11	Big-Data-Stream-Evaluation	240471	1
11	Big-Data-Stream-Evaluation	11840630	1
11	Big-Data-Stream-Evaluation	334639	1
11	Big-Data-Stream-Evaluation	13005410	1
11	Physical-Design	2225761	1
11	Physical-Design	6061813	1
11	Physical-Design	6065062	2
11	Physical-Design	15204051	1
11	Physical-Design	3413424	1
11	Physical-Design	10407760	1
11	Physical-Design	10745084	1
11	Physical-Design	10087396	1

11	Physical-Design	6169658	1
11	Physical-Design	4971437	1
11	Conceptual-Design	10641999	1
11	Conceptual-Design	1670332	1
11	Conceptual-Design	5074129	1
11	Conceptual-Design	600256	1
11	Conceptual-Design	4322819	1
11	Conceptual-Design	2567254	1
11	Conceptual-Design	280991	1
11	Conceptual-Design	11565906	1
11	Conceptual-Design	17401679	1
12	Middleware-Search	1919472	3
12	Middleware-Search	1112279	2
12	Middleware-Search	32851	3
12	Middleware-Search	309374	2
12	JSON-Search	15056878	2
12	JSON-Search	4633611	3
12	Messaging-Evaluation	3280676	3
12	Messaging-Evaluation	27666943	3
12	Messaging-Evaluation	9569851	3
12	Messaging-Evaluation	1919472	3
12	Big-Data-Stream-Evaluation	11840630	3
12	Big-Data-Stream-Evaluation	61437	3
12	Big-Data-Stream-Evaluation	6664445	3
12	Big-Data-Stream-Evaluation	10334306	2
12	Physical-Design	6889265	3
12	Physical-Design	2225761	1
12	Physical-Design	3921436	3
12	Physical-Design	7921324	3
12	Physical-Design	296650	3
12	Physical-Design	15204051	2
12	Conceptual-Design	13592894	3
12	Conceptual-Design	1670332	2
12	Conceptual-Design	16838416	2
13	Middleware-Search	1112279	3
13	Middleware-Search	2912051	3
13	Middleware-Search	2388569	2
13	Middleware-Search	3164821	3
13	Middleware-Search	12099446	1
13	Middleware-Search	24176969	1
13	Middleware-Search	2936598	1
13	Middleware-Search	12733985	2
13	Middleware-Search	29584618	1
13	Middleware-Search	2943935	1
13	Middleware-Search	417453	3

13	Middleware-Search	2912051	3
13	Middleware-Search	3164821	3
13	Middleware-Search	12099446	1
13	Middleware-Search	24176969	1
13	Middleware-Search	2936598	1
13	Middleware-Search	8349020	1
13	Middleware-Search	393996	3
13	Middleware-Search	3198781	1
13	Middleware-Search	4308343	1
13	Messaging-Evaluation	27666943	2
13	Messaging-Evaluation	21808529	1
13	Messaging-Evaluation	731233	3
13	Messaging-Evaluation	70441517	3
13	Messaging-Evaluation	12559570	1
13	Messaging-Evaluation	10445621	1
13	Messaging-Evaluation	12130481	1
13	Messaging-Evaluation	20520492	3
13	Messaging-Evaluation	23903843	1
13	Messaging-Evaluation	6664445	1
13	Messaging-Evaluation	20655367	1
13	Messaging-Evaluation	3280676	3
13	Messaging-Evaluation	9151698	2
13	Messaging-Evaluation	4287941	1
13	Messaging-Evaluation	6551718	3
13	Messaging-Evaluation	1164810	1
13	Messaging-Evaluation	14686136	1
13	Messaging-Evaluation	6933833	1
13	Messaging-Evaluation	14870832	1
13	Messaging-Evaluation	4405992	3
13	Big-Data-Stream-Evaluation	12559570	3
13	Big-Data-Stream-Evaluation	21808529	2
13	Big-Data-Stream-Evaluation	27666943	3
13	Big-Data-Stream-Evaluation	10445621	1
13	Big-Data-Stream-Evaluation	23903843	1
13	Big-Data-Stream-Evaluation	24553750	1
13	Big-Data-Stream-Evaluation	16284399	1
13	Big-Data-Stream-Evaluation	3792519	1
13	Big-Data-Stream-Evaluation	12130481	2
13	Big-Data-Stream-Evaluation	24287900	1
13	Big-Data-Stream-Evaluation	10513317	2
13	Big-Data-Stream-Evaluation	2336438	2
13	Big-Data-Stream-Evaluation	13681213	1
13	Big-Data-Stream-Evaluation	10375137	2
13	Big-Data-Stream-Evaluation	1307407	1
13	Big-Data-Stream-Evaluation	26623673	2

13	Big-Data-Stream-Evaluation	18566412	2
13	Big-Data-Stream-Evaluation	2679024	2
13	Big-Data-Stream-Evaluation	8744953	1
13	Big-Data-Stream-Evaluation	21786951	1
13	Big-Data-Stream-Evaluation	10513317	2
13	Big-Data-Stream-Evaluation	2336438	2
13	Big-Data-Stream-Evaluation	13681213	1
13	Big-Data-Stream-Evaluation	10375137	3
13	Big-Data-Stream-Evaluation	1307407	1
13	Big-Data-Stream-Evaluation	26623673	2
13	Big-Data-Stream-Evaluation	18566412	2
13	Big-Data-Stream-Evaluation	2679024	2
13	Big-Data-Stream-Evaluation	8744953	1
13	Big-Data-Stream-Evaluation	21786951	1
13	Conceptual-Design	3836900	2
13	Conceptual-Design	722675	1
13	Conceptual-Design	969964	1
13	Conceptual-Design	5044585	1
13	Conceptual-Design	2613348	3
13	Conceptual-Design	2503071	2
13	Conceptual-Design	10323393	1
13	Conceptual-Design	5407673	2
13	Conceptual-Design	476211	1
13	Conceptual-Design	10745084	1
13	Conceptual-Design	10641999	1
13	Conceptual-Design	1670332	1
13	Conceptual-Design	722675	1
13	Conceptual-Design	3836900	2
13	Conceptual-Design	4373833	1
13	Conceptual-Design	969964	3
13	Conceptual-Design	10323393	1
13	Conceptual-Design	11987838	1
13	Conceptual-Design	226953	1
13	Conceptual-Design	922116	1
13	Conceptual-Design	5296936	3
13	Conceptual-Design	6302341	2
13	Conceptual-Design	7882660	1
13	Conceptual-Design	11378046	3
13	Conceptual-Design	969964	3
13	Conceptual-Design	2567254	3
13	Conceptual-Design	698998	3
13	Conceptual-Design	162376	3
13	Conceptual-Design	57665	1
13	Conceptual-Design	17401679	1
13	Conceptual-Design	2554999	1

13	Conceptual-Design	3513112	1
13	Conceptual-Design	514306	1
13	Conceptual-Design	11755146	1
13	Conceptual-Design	12349690	2
13	Conceptual-Design	16482269	1
13	Conceptual-Design	2508361	1
13	Conceptual-Design	3709987	1
13	Conceptual-Design	2567254	3
13	Conceptual-Design	3543835	1
14	Middleware-Search	1112279	1
14	Middleware-Search	13483809	3
14	Middleware-Search	14713711	2
14	Middleware-Search	9285953	3
14	Middleware-Search	15159972	3
14	Messaging-Evaluation	22989833	3
14	Messaging-Evaluation	7044157	3
14	Messaging-Evaluation	4971437	2
14	Messaging-Evaluation	12130481	2
14	Messaging-Evaluation	27666943	3
14	Messaging-Evaluation	2705043	3
14	Messaging-Evaluation	2124221	3
14	Big-Data-Stream-Evaluation	2486721	1
14	Big-Data-Stream-Evaluation	1809296	3
14	Big-Data-Stream-Evaluation	12130481	3
14	Big-Data-Stream-Evaluation	27666943	2
14	Big-Data-Stream-Evaluation	13681213	2
14	Big-Data-Stream-Evaluation	3198781	3
14	Conceptual-Design	822328	1
14	Conceptual-Design	36999	3
14	Conceptual-Design	6349375	3
14	Conceptual-Design	3767660	2
14	Conceptual-Design	5082454	2
14	Conceptual-Design	5181419	1
15	JSON-Search	7410040	1
15	JSON-Search	2782076	2
15	JSON-Search	4697740	1
15	JSON-Search	2429844	1
15	JSON-Search	2782076	2
15	JSON-Search	7410040	1
15	JSON-Search	1807617	1
15	JSON-Search	7425808	1
15	Messaging-Evaluation	2124221	2
15	Messaging-Evaluation	20740114	2
15	Messaging-Evaluation	6104418	2
15	Messaging-Evaluation	1823705	3

15	Messaging-Evaluation	27666943	3
15	Messaging-Evaluation	7044157	3
15	Messaging-Evaluation	12130481	3
15	Messaging-Evaluation	17708489	3
15	Messaging-Evaluation	27666943	3
15	Messaging-Evaluation	7044157	3
15	Messaging-Evaluation	12130481	3
15	Messaging-Evaluation	17708489	3
15	Messaging-Evaluation	22989833	3
15	Messaging-Evaluation	7044157	3
15	Messaging-Evaluation	6636213	2
15	Messaging-Evaluation	17806977	2
15	Big-Data-Stream-Evaluation	2486721	1
15	Big-Data-Stream-Evaluation	3916983	1
15	Big-Data-Stream-Evaluation	334639	1
15	Big-Data-Stream-Evaluation	3102551	1
15	Big-Data-Stream-Evaluation	12559570	3
15	Big-Data-Stream-Evaluation	3921436	2
15	Big-Data-Stream-Evaluation	3792519	2
15	Big-Data-Stream-Evaluation	6930236	2
15	Big-Data-Stream-Evaluation	773503	3
15	Big-Data-Stream-Evaluation	240471	2
15	Big-Data-Stream-Evaluation	12634965	3
15	Big-Data-Stream-Evaluation	4311279	3
15	Big-Data-Stream-Evaluation	3034054	2
15	Big-Data-Stream-Evaluation	8845186	2
15	Big-Data-Stream-Evaluation	9623482	2
15	Big-Data-Stream-Evaluation	3921436	2
15	Big-Data-Stream-Evaluation	3792519	2
15	Big-Data-Stream-Evaluation	10055290	1
15	Physical-Design	11545895	1
15	Physical-Design	7874562	1
15	Physical-Design	13965635	2
15	Physical-Design	1603938	1
15	Physical-Design	12008	2
15	Physical-Design	11545895	1
15	Physical-Design	7874562	1
15	Physical-Design	13965635	2
15	Physical-Design	1603938	1
15	Physical-Design	16994238	1
15	Physical-Design	10347751	2
15	Physical-Design	11545895	1
15	Physical-Design	7874562	1
15	Physical-Design	13965635	2
15	Physical-Design	1603938	1

15	Physical-Design	16994238	1
15	Physical-Design	11545895	1
15	Physical-Design	7874562	1
15	Physical-Design	13965635	2
15	Conceptual-Design	250207	1
15	Conceptual-Design	11013802	3
15	Conceptual-Design	6663888	1
15	Conceptual-Design	1665976	1
15	Conceptual-Design	7682662	1
15	Conceptual-Design	447518	3
15	Conceptual-Design	2023027	3
15	Conceptual-Design	3051326	3
15	Conceptual-Design	3488395	3
15	Conceptual-Design	377656	2
15	Conceptual-Design	2503071	3
15	Conceptual-Design	309374	2
15	Conceptual-Design	13879379	2
15	Conceptual-Design	773503	2
15	Conceptual-Design	11013802	3
15	Conceptual-Design	4163066	2
15	Conceptual-Design	561915	2
16	JSON-Search	57689	3
16	JSON-Search	26380184	1
16	Messaging-Evaluation	27666943	3
16	Messaging-Evaluation	731233	3
16	Messaging-Evaluation	12130481	2
16	Physical-Design	11248510	2
16	Physical-Design	2225761	3
16	Physical-Design	18531072	3
16	Conceptual-Design	10641999	3
16	Conceptual-Design	1670332	3
16	Conceptual-Design	5074129	1
16	Conceptual-Design	2554999	1
16	Conceptual-Design	2508361	3
16	Big-Data-Stream-Evaluation	12559570	2
16	Big-Data-Stream-Evaluation	10513317	2
16	Big-Data-Stream-Evaluation	2336438	1

List of Figures

1.1 Summary of research questions and contributions 7

1.2 High-level overview on the research process 8

1.3 Thesis outline 13

1.4 Thesis contributions in each chapter and their relationships 14

2.1 The structure of a Stack Overflow post. The dotted rectangles represent regions in a Stack Overflow post. Bubbles specify elements of a post, and their associated region. 22

3.1 Research Process Diagram. 36

3.2 An example for a technology features' tree. 40

3.3 A subset from a technology features' tree with both feature and concern based drawbacks (ASTA). 43

3.4 Software Architecture Knowledge (AK) Metamodel. 45

3.5 An example showing the interaction of objects based on the proposed architectural knowledge domain model 50

4.1 Research Process Diagram. 54

4.2 Distribution of ARP types according to purpose and solution types. 60

4.3 Agreement and confidence for ARP types according to the purpose of the question. 75

4.4 Agreement and confidence for ARP types according to the solution type. 75

5.1 An example for an annotated statement in ARP 80

5.2 Examples of annotated sentences of architecture configuration (CONF) ontology class. Each sentence is further annotated with its composing ontology classes. The three sentences belong to three posts: stackoverflow.com/questions/12783677, stackoverflow.com/questions/4473567, and stackoverflow.com/questions/19758215 83

5.3 Examples of annotated sentences of component behavior (CB) ontology class. Each sentence is further annotated with its composing ontology classes. The three sentences belong to three posts: stackoverflow.com/questions/1582952, stackoverflow.com/questions/380052, and stackoverflow.com/questions/4473567 84

5.4 Example for an annotated sentences of existing system (EX) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belong to the post: stackoverflow.com/questions/13016406 85

5.5 Example for an annotated sentences of design issue (DI) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belong to the post: stackoverflow.com/questions/4741713 85

5.6 Example for an annotated sentences of the technology feature requirement, a subtype from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/14342430 86

5.7 Example for an annotated sentences of the quality attribute requirement, a subtype from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/2567254 86

5.8 Example for an annotated sentences of the team skills constraint, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/1426249 86

5.9 Example for an annotated sentences of the development time constraint, a subtype from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/807962 87

5.10 Example for an annotated sentences of the solution constraint, a sub-type from the requirements and constraints (REQ) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/12783677 87

5.11 Example for an annotated sentences of the user request (UR) ontology class. The sentence is further annotated with its composing ontology classes. The sentences belong to the posts: stackoverflow.com/questions/1426249 and stackoverflow.com/questions/10621648 88

5.12 Example for an annotated sentences of the development feature, a sub-type from the technology feature (FEAT) ontology class. The sentence is further annotated with its composing ontology classes. The sentence belongs to the post: stackoverflow.com/questions/1429318 88

5.13 Example for an annotated sentences of the behavioral features, a sub-type from the technology feature (FEAT) ontology class. The sentences are further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/10375137, stackoverflow.com/questions/4473567, and stackoverflow.com/questions/7583132 89

5.14 Example for an annotated sentences of the development ASTA, a sub-type from the technology benefits and drawbacks (ASTA) ontology class. The sentence is further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/100993, stackoverflow.com/questions/807692, and stackoverflow.com/questions/1582952 90

5.15 Example for an annotated sentences of the behavioral ASTA, a sub-type from the technology benefits and drawbacks (ASTA) ontology class. The sentences are further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/4473567, stackoverflow.com/questions/5145129, and stackoverflow.com/questions/1426249 90

5.16 Examples for annotated sentences of the ADD. The sentences are further annotated with their composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/10156388 and stackoverflow.com/questions/9535421 91

5.17 Example for an annotated sentences of DR. The sentences are further annotated with its composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/1426249, stackoverflow.com/questions/1582952, and stackoverflow.com/questions/100993 92

5.18 Examples for annotated sentences of CASE. The sentences are further annotated with their composing ontology classes. The sentences belongs to posts: stackoverflow.com/questions/1429318, stackoverflow.com/questions/2508361, and stackoverflow.com/questions/5145129 93

5.19 The percentages of occurrences for composite ontology classes within the question and answers sections of a Stack Overflow post 94

5.20 An example for an annotated ARP https://stackoverflow.com/questions/4473567 96

5.21 The relative occurrences of ontology classes in each type of ARP 99

6.1 The main elements of a scenario to search for architecture knowledge. 109

6.2 Relating information intensive architecture design activities (in gray) with ISTs . 110

7.1 Example of multi-ontology-class preprocessing 120

7.2 Comparing F-scores across different classification approaches and types of posts in the 2-class dataset 123

7.3 Comparing F-scores across different classification approaches and types of posts in the 4-class dataset 123

7.4 Comparing F-scores across different classification approaches and types of posts in the 2-class dataset using the development sample as training data and testing sample for testing the generalizability of the classification approaches 126

7.5 Comparing F-scores across different classification approaches and types of posts in the 4-class dataset using the development sample as training data and testing sample for testing the generalizability of the classification approaches 126

8.1 Enhanced search process (gray elements are the extensions of a keyword-based search) 134

8.2 Number of relevant posts identified by participants for each design step. 141

8.3 nDCG@1-3 for NORMAL and ENHANCED. The lower and upper boxes are the lower and upper quartiles of the distribution, respectively. The vertical thin line is from the minimum to the maximum nDCG values. 141

8.4 Average Precision@k for each design activity 142

8.5 Average nDCG@k for each design activity 142

8.6 Significance T-value for nDCG@k and Precision@k. The Critical T-value is 1.963 (T-values > 1.963 are considered significant). The higher the T-value, the higher the probability for being significant. 143

A.1 The query used to gather stack overflow posts 157

A.2 Atlas.ti interface for documents coding 194

A.3 Atlas.ti interface for managing codes 194

A.4 Atlas.ti interface for managing documents 194

List of Tables

3.1 Interview Participants Experience Overview 37

3.2 Capabilities' Types and Architectural Concerns Relationships 41

3.3 Interview Data Analysis Results: *++: Concept Contribution, ✔: Concept Supported, Y: Concept Accepted, N: Concept in Doubt, –: No Answer Provided* . . . 49

4.1 Background of participants 57

4.2 Example for technology identification ARP, stackoverflow.com/questions/4473567 62

4.3 Example for technology feature identification ARP, stackoverflow.com/questions/1935040 62

4.4 Example for architecture configuration identification ARP, stackoverflow.com/questions/2897513 62

4.5 Example for technology evaluation ARP, stackoverflow.com/questions/32851 64

4.6 Example for technology feature evaluation ARP, stackoverflow.com/questions/5407673 64

4.7 Example for architecture configuration evaluation ARP, stackoverflow.com/questions/2498796 64

4.8 Example for the Explicit Technology Solutions Searching ARP Variation, stackoverflow.com/questions/7837189 65

4.9 Example for the Implicit Technology Solutions Searching ARP Variation, where the solution is a technology bundle, stackoverflow.com/questions/3400139 65

4.10 Example for the Implicit Technology Solutions Searching ARP Variation, where the solution is a technology feature, stackoverflow.com/questions/393580 65

4.11 Example for the Alternative Technology Solutions Searching ARP Variation, stackoverflow.com/questions/10156388 66

4.12 Example for the Solutions Recommendation for Requirements and Implementation of Conceptual Design ARP Variation, where the recommended solution is a technology bundle, stackoverflow.com/questions/2675793 66

4.13 Example for the Solutions Recommendation for Requirements and Implementation of Conceptual Design ARP Variation, where the recommended solution is an architecture configuration, stackoverflow.com/questions/3198781 . 67

4.14 Example for the Technology Independent Architectural Configuration Searching ARP Variation, stackoverflow.com/questions/1139203 67

4.15 Example for comparing solutions variation for architectural configurations, stackoverflow.com/questions/17814436 68

4.16 Example for context independent solution assessment variation for technology bundle, stackoverflow.com/questions/512807 68

4.17 Example for context independent solution assessment variation for technology feature, stackoverflow.com/questions/1189420 68

4.18 Example for solution scenarios and context variation for technology bundle, stackoverflow.com/questions/4499510 69

4.19 Example for solution scenarios and context variation for architecture configuration, stackoverflow.com/questions/969964 69

4.20 Example for the "context dependent solution assessment" variation among a technology bundle, stackoverflow.com/questions/9502548 69

4.21 Example for the "context dependent solution assessment" variation among a technology feature, stackoverflow.com/questions/3543835 70

4.22 Example for the "context dependent solution assessment" variation among a technology bundle, stackoverflow.com/questions/10160463 70

4.23 Example for the technology solutions interoperability assessment variation, stackoverflow.com/questions/2669573 72

4.24 Example for the solution definition and analysis variation for technology bundle, stackoverflow.com/questions/3355082 72

4.25 Example for the solution definition and analysis variation for architecture configuration, stackoverflow.com/questions/3055713 72

4.26 Agreement and confidence for each participant 74

5.1 *Simple AK Concept* ontology classes 82

5.2 *Lexical Trigger* ontology classes 83

5.3 Statistically significant composing ontology classes for composite ontology classes 97

5.4 Number of annotations for each ontology class among the different types of ARPs 98

6.1 Interview participants 104

6.2 Benefits, problems and solutions mapped to interviewees 108

6.3 Information searching activities (ISAs) for AK in developer communities 111

7.1 Top tags with highest differences in occurrences between Stack Overflow programming posts and ARPs . . . 117

7.2 Comparison of Classification Approaches using the Bag-of-Words . . . 122

7.3 Comparison of Classification Approaches using the Single-Ontology-Class . . . 122

7.4 Comparison of Classification Approaches using the Multi-Ontology-Class . . . 122

7.5 Comparison of the Best Performing Classification Approaches and their Combination using the Ensemble Learning . . . 122

7.6 Confusion matrix for classification approaches . . . 124

7.7 Comparing F-scores across different classification approaches using the development sample as training data and testing sample for testing the generalizability of the classification approaches . . . 125

7.8 Top 30 distinctive terms between the ARPs and PPPs in our sample . . . 129

7.9 Features (Based on Ontology Classes) with Highest IGR . . . 130

7.10 Features (Based on Sequences of Ontology Classes) with Highest IGR and Their Mapping to Composite Ontology Classes . . . 131

8.1 Frequency of annotations, which belong to relevant ontology classes in different types of posts . . . 135

8.2 Experiences of Practitioners in Experiment . . . 137

Bibliography

[AF00] Eric C. Adams and Christopher Freeman. Communities of practice: bridging technology and knowledge assessment. *Journal of Knowledge Management*, 4(1):38–44, 2000.

[ANGB+05] T. Al-Naeem, I. Gorton, M.A. Babar, F. Rabhi, and B. Benatallah. A quality-driven systematic approach for architecting distributed software applications. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 244–253, May 2005.

[ASR17] R. Abdalkareem, E. Shihab, and J. Rilling. What do developers use the crowd for? a study using stack overflow. *IEEE Software*, 34(2):53–60, 2017.

[Aud10] R. Audi. *Epistemology: A Contemporary Introduction to the Theory of Knowledge*. Epistemology: A Contemporary Introduction to the Theory of Knowledge. Routledge, 2010.

[AZ05] Paris Avgeriou and Uwe Zdun. Architectural patterns revisited – a pattern language. In *Proceedings 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee*, pages 1–39, 2005.

[AZ14] Mohsen Anvaari and Olaf Zimmermann. Semi-automated design guidance enhancer (sadge): A framework for architectural guidance development. In Paris Avgeriou and Uwe Zdun, editors, *Software Architecture*, pages 41–49, Cham, 2014. Springer International Publishing.

[AZE+07] A. Arsanjani, L. J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah. S3: A service-oriented reference architecture. *IT Professional*, 9(3):10–17, May 2007.

[BCK03] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[BCK12] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, 3rd edition, 2012.

[BDTD18] Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl. Sotorrent: Reconstructing and analyzing the evolution stack overflow posts. In *ICSE'18: The 40th International Conference on Software Engineering*, May 27-June 3, 2018 2018.

[BH05] Katriina BystrÃűm and Preben Hansen. Conceptual framework for tasks in information studies. *Journal of the American Society for Information Science and Technology*, 56(10):1050–1061, 2005.

[BHS07a] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Wiley, Chichester, UK, 2007.

[BHS07b] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages*. Wiley, Chichester, UK, 2007.

[BHS13] B. Bazelli, A. Hindle, and E. Stroulia. On the personality traits of stackoverflow users. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 460–463, Sept 2013.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BMPP18] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. Automatically classifying posts into question categories on stack overflow. In *IEEE International Conference on Program Comprehension, ICPC*, 2018.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, 1 edition, July 1996.

[Boe06] Barry Boehm. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 12–29, New York, NY, USA, 2006. ACM.

[Bor00] Pia Borlund. Experimental components for the evaluation of interactive information retrieval systems. *Journal of Documentation*, 56(1):71–90, 2000.

[Bor03] Pia Borlund. The iir evaluation model: a framework for evaluation of interactive information retrieval systems. *Information Research*, 8(3), 2003.

[Bos00] Jan Bosch. *Design and use of software architectures: Adopting and evolving a product-line approach*. ACM Press/Addison-Wesley, New York, NY, USA, 2000.

[Bos16] Jan Bosch. Speed, data, and ecosystems: the future of software engineering. *IEEE Software*, 33(1):82–88, 2016.

[BR10] Stephan Bode and Matthias Riebisch. Impact evaluation for quality-oriented architectural decisions regarding evolvability. In Muhammad Babar and Ian Gorton, editors, *Proceedings 4th European Conference on Software Architecture, ECSA 2010*, volume 6285 of *LNCS*, pages 182–197. Springer Berlin / Heidelberg, 2010.

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[BSB+17] Manoj Bhat, Klym Shumaiev, Andreas Biesdorf, Uwe Hohenstein, and Florian Matthes. Automatic extraction of design decisions from issue management systems: A machine learning based approach. In Antónia Lopes and Rogério de Lemos, editors, *Software Architecture*, pages 138–154, Cham, 2017. Springer International Publishing.

[BSK+18] Manoj Bhat, Klym Shumaiev, Kevin Koch, Uwe Hohenstein, Andreas Biesdorf, and Florian Matthes. An expert recommendation system for design decision making who should be involved in making a design decision? In *IEEE International Conference on Software Architecture, ICSA*, 2018.

[BTH14] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Softw. Engg.*, 19(3):619–654, June 2014.

[BW99] W.R. Bukowitz and R.L. Williams. *The knowledge management fieldbook*. Financial Times Prentice Hall London, 1999.

[BWG05] Muhammad Ali Babar, Xiaowen Wang, and Ian Gorton. PAKME: A tool for capturing and using architecture design knowledge. In *2005 INMIC*. IEEE, 2005.

[CJT+16] Rafael Capilla, Anton Jansen, Antony Tang, Paris Avgeriou, and Muhammad Ali Babar. 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, 116(Supplement C):191 – 205, 2016.

[CLvV07] Viktor Clerc, Patricia Lago, and Hans van Vliet. The architect's mindset. In Sven Overhage, ClemensA. Szyperski, Ralf Reussner, and JudithA. Stafford, editors, *Software Architectures, Components, and Applications*, volume 4880 of *Lecture Notes in Computer Science*, pages 231–249. Springer Berlin Heidelberg, 2007.

[CNPDn06] Rafael Capilla, Francisco Nava, Sandra Pérez, and JC Dueñas. A web-based tool for managing architectural design decisions. *SIGSOFT Softw. Eng. Notes*, 31(5), 2006.

[Coh60] J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

[CP01] Don Cohen and Laurence Prusak. *In good company*. Harvard Business School Press, Boston, 2001.

[CS10] M. Cade and H. Sheil. *Sun Certified Enterprise Architect for Java EE Study Guide*. Pearson Education, 2010.

[CVEK13] H. Cervantes, P. Velasco-Elizondo, and R. Kazman. A principled way to use frameworks in architecture design. *Software, IEEE*, 30(2):46–53, March 2013.

[CZZK11] Rafael Capilla, Olaf Zimmermann, Uwe Zdun, and Jochen M. Küster. An enhanced architectural knowledge metamodel linking architectural design decisions to other artifacts in the software engineering lifecycle. In *Software Architecture*, pages 303–318. Springer Berlin Heidelberg, 2011.

[Dal11] Kimiz Dalkir. *Knowledge Management in Theory and Practice*. The MIT Press, 2011.

[dBFL+07] Remco C. de Boer, Rik Farenhorst, Patricia Lago, Hans van Vliet, Viktor Clerc, and Anton Jansen. *Architectural Knowledge: Getting to the Core*, pages 197–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[dBvV11] R. C. d. Boer and H. v. Vliet. Experiences with semantic wikis for architectural knowledge management. In *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, pages 32–41, June 2011.

[DGLT+14a] K. A. De Graaf, P. Liang, A. Tang, W. R. Van Hage, and H. Van Vliet. An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7):1053–1064, 2014.

[dGLT+14b] K.A. de Graaf, P. Liang, A. Tang, W.R. van Hage, and H. van Vliet. An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7):1053 – 1064, 2014.

[Dix91] R.M.W. Dixon. *A new approach to English grammar, on semantic principles*. Oxford University Press, Incorporated, 1991.

[DPP97] Thomas H. Davenport, Lawrence Prusak, and Laurence Prusak. *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, Boston, MA, USA, 1997.

[dSCM14] Lucas B. L. de Souza, Eduardo C. Campos, and Marcelo de A. Maia. Ranking crowd knowledge to assist software development. In *Proceedings of the 22Nd International Conference on Program Comprehension*, ICPC 2014, pages 72–82, New York, NY, USA, 2014. ACM.

[FBC+13] Davide Falessi, Lionel C. Briand, Giovanni Cantone, Rafael Capilla, and Philippe Kruchten. The value of design rationale information. *ACM Trans. Softw. Eng. Methodol.*, 22(3):21:1–21:32, July 2013.

[FCK07] D. Falessi, G. Cantone, and P. Kruchten. Do architecture design methods meet architects' needs? In *Software Architecture, 2007. WICSA '07. The Working IEEE/IFIP Conference on*, pages 5–5, Jan 2007.

[FCK08] D. Falessi, G. Cantone, and P. Kruchten. Value-based design decision rationale documentation: Principles and empirical feasibility study. In *Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on*, pages 189–198, Feb 2008.

[FCKK11] Davide Falessi, Giovanni Cantone, Rick Kazman, and Philippe Kruchten. Decision-making techniques for software architecture design: A comparative survey. *ACM Comput. Surv.*, 43(4):33:1–33:28, October 2011.

[FCL+11] Miriam FernÃąndez, IvÃąn Cantador, Vanesa LÃşpez, David Vallet, Pablo Castells, and Enrico Motta. Semantically enhanced information retrieval: An ontology-based approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):434 – 452, 2011.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, 1997.

[FGWY07] J. Fang, L. Guo, X. Wang, and N. Yang. Ontology-based automatic classification and ranking for web documents. In *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, volume 3, pages 627–631, Aug 2007.

[FILvV08] R. Farenhorst, R. Izaks, P. Lago, and H. v. Vliet. A just-in-time architectural knowledge sharing portal. In *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, pages 125–134, Feb 2008.

[FVC06] Miriam Fernández, David Vallet, and Pablo Castells. Probabilistic score normalization for rank aggregation. In *Advances in Information Retrieval*, pages 553–556. Springer, 2006.

[FvKSJ04] U. Flick, E. von Kardoff, I. Steinke, and B. Jenner. *A Companion to Qualitative Research*. SAGE Publications, 2004.

[GBR15] L. Guerrouj, D. Bourque, and P. C. Rigby. Leveraging informal documentation to summarize classes and methods in context. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 639–642, May 2015.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Softwaresystemen*. Addison-Wesley Professional, 1994.

[GJG04] Laura A. Granka, Thorsten Joachims, and Geri Gay. Eye-tracking analysis of user behavior in www search. In *International Conference on Research and Development in Information Retrieval*, pages 478–479. ACM, 2004.

[GKN15] I. Gorton, J. Klein, and A. Nurgaliev. Architecture knowledge for evaluating scalable databases. In *WICSA, IEEE/IFIP*, pages 95–104, May 2015.

[GLB03] I. Gorton, A. Liu, and P. Brebner. Rigorous evaluation of cots middleware technology. *Computer*, 36(3):50–55, Mar 2003.

[Gle05] B. Gleason. *The development of language*. Pearson Education, 2005.

[GLJ11] Swapna Gottipati, David Lo, and Jing Jiang. Finding relevant answers in software forums. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 323–332. IEEE, 2011.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199 – 220, 1993.

[GXY+17] I. Gorton, R. Xu, Y. Yang, H. Liu, and G. Zheng. Experiments in curation: Towards machine-assisted construction of software architecture knowledge bases. In *IEEE/IFIP International Conference on Software Architecture (ICSA)*, pages 79–88. IEEE, 2017.

[HA97] John Hagel, III and Arthur G. Armstrong. *Net Gain: Expanding Markets Through Virtual Communities*. Harvard Business School Press, Boston, MA, USA, 1997.

[HA05] S.E. Hove and B. Anda. Experiences from conducting semi-structured interviews in empirical software engineering research. In *Software Metrics, 2005. 11th IEEE International Symposium*, pages 10 pp.–23, Sept 2005.

[HA11] Uwe van Heesch and Paris Avgeriou. Mature architecting - a survey about the reasoning process of professional architects. In *WICSA IEEE/IFIP*, pages 260–269, 2011.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[HKN+07] Christine Hofmeister, Philippe Kruchten, Robert L. Nord, Henk Obbink, Alexander Ran, and Pierre America. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, 80(1):106–126, Jan 2007.

[IM02] Rus I. and Lindvall M. Knowledge management in software engineering. *IEEE Software*, 21(6):26–38, May.-June. 2002.

[Int11] International Standardization Organisation. Iso/iec/ieee 42010 - systems and software engineering - architecture description, 2011.

[ISO01] ISO/IEC. *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001.

[Jac76] R. Jackendoff. *Toward an Explanatory Semantic Representation*. MIT Press, 1976.

[JB05] Anton Jansen and Jan Bosch. Software architecture as a set of architectural design decisions. In *WICSA*, pages 109–120, 2005.

[JDAH07] Anton Jansen, Jan Der Ven, Paris Avgeriou, and Dieter Hammer. Tool Support for Architectural Decisions. In *WICSA'07*, pages 4–4. IEEE, January 2007.

[JdVAvV08] Anton Jansen, Tjaard de Vries, Paris Avgeriou, and Martijn van Veelen. Sharing the architectural knowledge of quantitative analysis. In Steffen Becker, Frantisek Plasil, and Ralf Reussner, editors, *Quality of Software Architectures. Models and Architectures*, pages 220–234, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[JL95] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[KC16] Rick Kazman and Humberto Cervantes. *Designing Software Architectures: A Practical Approach*. Addison-Wesley Professional, 2016.

[KHDM98] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[KLV06] Philippe Kruchten, Patricia Lago, and Hans Vliet. Building up and reasoning about architectural knowledge. In Christine Hofmeister, Ivica Crnkovic, and Ralf Reussner, editors, *Quality of Software Architectures*, volume 4214 of *Lecture Notes in Computer Science*, pages 43–58. Springer Berlin Heidelberg, 2006.

[Koh95] Ron Kohavi. *The power of decision tables*. Springer Berlin Heidelberg, 1995.

[Kru95] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Softw.*, 12(6):42–50, November 1995.

[Kru15] P. Kruchten. Lifelong learning for lifelong employment. *IEEE Software*, 32(4):85–87, July 2015.

[Kun04] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[LAH10] P. Lago, P. Avgeriou, and R. Hilliard. Guest editors' introduction: Software architecture: Framing stakeholders' concerns. *Software, IEEE*, 27(6):20–24, Nov 2010.

[LCAC12] Claudia LÃşpez, VÃ■ctor Codocedo, HernÃąn Astudillo, and Luiz Marcio Cysneiros. Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach. *Science of Computer Programming*, 77(1):66 – 80, 2012. System and Software Solution Oriented Architectures.

[LHF05] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Mach. Learn.*, 59(1-2):161–205, May 2005.

[LLA16] G. A. Lewis, P. Lago, and P. Avgeriou. A decision model for cyber-foraging systems. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 51–60, April 2016.

[LTZ12] Ioanna Lytra, Huy Tran, and Uwe Zdun. Constraint-based consistency checking between design decisions and component models for supporting software architecture evolution. In Tom Mens, Anthony Cleve, and Rudolf Ferenc, editors, *16th European Conference on Software Maintenance and Reengineering, CSMR 2012, Szeged, Hungary, March 27-30, 2012*, pages 287–296. IEEE, 2012.

[LZ13] Ioanna Lytra and Uwe Zdun. Supporting architectural decision making for systems-of-systems design under uncertainty. In *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems, Montpellier, France, July 2, 2013*, pages 43–46. ACM, 2013.

[May14] P. Mayring. *Qualitative Content Analysis. Theoretical Foundation, Basic Procedures and Software Solution*. Beltz, 2014.

[MBF+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[MBH+15] Laura Moreno, Gabriele Bavota, Sonia Haiduc, Massimiliano Di Penta, Rocco Oliveto, Barbara Russo, and Andrian Marcus. Query-based configuration of text retrieval solutions for software engineering tasks. In *10th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 567–578. ACM, 2015.

[McE02] Mark W. McElroy. *The New Knowledge Management: Complexity,Learning,and Sustainable Innovation*. Butterworth-Heinemann, Newton, MA, USA, 2002.

[MHvHA11] Sara Mahdavi-Hezavehi, Uwe van Heesch, and Paris Avgeriou. A pattern language for architecture patterns and software technologies introducing technology pattern languages. In *Proceedings of the 16th EuroPLoP*. Conference Proceedings, 2011.

[Mon06] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2006.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[MT00] Nenad Medvidovic and Richard N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Trans. Softw. Eng.*, 26(1):70–93, January 2000.

[MTA+16] Christian Manteuffel, Dan Tofan, Paris Avgeriou, Heiko Koziolek, and Thomas Goldschmidt. Decision architect âĂŞ a decision documentation tool for industry. *Journal of Systems and Software*, 112:181 – 198, 2016.

[MTK+14] Christian Manteuffel, Dan Tofan, Heiko Koziolek, Thomas Goldschmidt, and Paris Avgeriou. Industrial Implementation of a Documentation Framework for Architectural Decisions. In *WICSA'14*, pages 225–234. IEEE, 2014.

[MW12] C. Miesbauer and R. Weinreich. Capturing and maintaining architectural knowledge using context information. In *WICSA/ECSA, IEEE/IFIP*, pages 206–210, Aug 2012.

[MW13] Cornelia Miesbauer and Rainer Weinreich. Classification of design decisions: An expert survey in practice. In *Proceedings of the 7th European Conference on Software Architecture*, ECSA'13, pages 130–145, Berlin, Heidelberg, 2013. Springer-Verlag.

[NP13] Marcin Nowak and Cesare Pautasso. Team Situational Awareness and Architectural Decision Making with the Software Architecture Warehouse. In *ECSA'13*. IEEE, 2013.

[NSMB12a] S.M. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example?: A study of programming q and a in stackoverflow. In *ICSM*, pages 25–34, Sept 2012.

[NSMB12b] S.M. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example?: A study of programming q and a in stackoverflow. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 25–34, Sept 2012.

[Ome01] Borys Omelayenko. Learning of ontologies for the web: the analysis of existent approaches. In *First International Workshop on Web Dynamics in Conjunction with the Eighth International Conference on Database Theory London, UK*, page 16, 2001.

[PBDP+14] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. Mining stackoverflow to turn the ide into a self-confident programming prompter. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 102–111, New York, NY, USA, 2014. ACM.

[PBL13a] L. Ponzanelli, A. Bacchelli, and M. Lanza. Leveraging crowd knowledge for software comprehension and development. In *2013 17th European Conference on Software Maintenance and Reengineering*, pages 57–66, March 2013.

[PBL13b] L. Ponzanelli, A. Bacchelli, and M. Lanza. Seahawk: Stack overflow in the ide. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1295–1298, May 2013.

[Pea00] K. Pearson. On a criterion that a given system of deviations from the probable in the case of correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. pages 157–175, 1900.

[Phi99] Geoffrey Phipps. Comparing observed bug and productivity rates for java and c++. *Software: Practice and Experience*, 29(4):345–358, 1999.

[Pla99] John C. Platt. Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[PM13] Dennis Pagano and Walid Maalej. How do open source communities blog? *Empirical Software Engineering*, 18(6):1090–1124, 2013.

[Pol66] Michael Polanyi. *The Tacit Dimension*. Doubleday, New York, 1966.

[PT09] Microsoft Patterns and Practices Team. *Microsoft Application Architecture Guide, 2nd Edition*. Microsoft Press, 2009.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[RM14] V. S. Rekhav and H. Muccini. A study on group decision-making in software architecture. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 185–194, April 2014.

[Rok10] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39, 2010.

[RR08] R.L. Rosnow and R. Rosenthal. *Beginning Behavioral Research: A Conceptual Primer*. Pearson/Prentice Hall, 2008.

[RS15] Christoffer Rosen and Emad Shihab. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, pages 1–32, 2015.

[RYR14] M. M. Rahman, S. Yeasmin, and C. K. Roy. Towards a context-aware ide-based meta search engine for recommendation about programming errors and exceptions. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pages 194–203, Feb 2014.

[Sch05] Karin Kipper Schuler. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. PhD thesis, Philadelphia, PA, USA, 2005. AAI3179808.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[Sei06] I. Seidman. *Interviewing as Qualitative Research: A Guide for Researchers in Education and the Social Sciences*. Teachers College Press, 2006.

[SLK09] Mojtaba Shahin, Peng Liang, and Mohammad Reza Khayyambashi. Architectural design decision: Existing models and tools. In *WICSA/ECSA'09*. IEEE, 2009.

[SM86] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[Squ15] M. Squire. Should we move to stack overflow? measuring the utility of social media for developer support. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 219–228, May 2015.

[SRZ15] M. Soliman, M. Riebisch, and U. Zdun. Enriching architecture knowledge with technology design decisions. In *WICSA*, pages 135–144, May 2015.

[Sut92] Jean Tague Sutcliffe. The pragmatics of information retrieval experimentation, revisited. *Information Processing and Management*, 28(4):467 – 490, 1992.

[SZP07] Nelly Schuster, Olaf Zimmermann, and Cesare Pautasso. Adkwik: Web 2.0 collaboration system for architectural decision engineering. In *Nineteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2007)*, pages 255–260, Boston, USA, July 2007.

[TA05] J. Tyree and A. Akerman. Architecture Decisions: Demystifying Architecture. *IEEE Software*, 22(2):19–27, 2005.

[TAJ+10] Antony Tang, Paris Avgeriou, Anton Jansen, Rafael Capilla, and Muhammad Ali Babar. A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, 83(3):352–370, 2010.

[Tan11] Antony Tang. Software designers, are you biased? In *Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge*, SHARK '11, pages 1–8, New York, NY, USA, 2011. ACM.

[TBGH05] A. Tang, M. A. Babar, I. Gorton, and Jun Han. A survey of the use and documentation of architecture design rationale. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pages 89–98, 2005.

[TBS11a] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807, New York, NY, USA, 2011. ACM.

[TBS11b] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *ICSE*, pages 804–807. ACM, 2011.

[TBSvdW18] Antony Tang, Floris Bex, Courtney Schriek, and Jan Martijn E.M. van der Werf. Improving software design reasoningâĂŞa reminder card approach. *Journal of Systems and Software*, 144:22 – 40, 2018.

[TGA11] Dan Tofan, Matthias Galster, and Paris Avgeriou. Capturing tacit architectural knowledge using the repertory grid technique (nier track). In *ICSE*, pages 916–919. ACM, 2011.

[TGAS14] Dan Tofan, Matthias Galster, Paris Avgeriou, and Wes Schuitema. Past and future of software architectural decisions - a systematic mapping study. *Information and Software Technology*, 56:850 – 872, 2014.

[THV09] A. Tang, J. Han, and R. Vasa. Software architecture design reasoning: A case for improved methodology support. *IEEE Software*, 26(2):43–49, March 2009.

[TMD09] R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing, 2009.

[TR16] Christoph Treude and Martin P. Robillard. Augmenting api documentation with insights from stack overflow. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 392–403, New York, NY, USA, 2016. ACM.

[TTHV08] Antony Tang, Minh H. Tran, Jun Han, and Hans Vliet. Design reasoning improves software design quality. In *Proceedings of the 4th International Conference on Quality of Software-Architectures: Models and Architectures*, QoSA '08, pages 28–42, Berlin, Heidelberg, 2008. Springer-Verlag.

[TVV09] A. Tang and H. Van Vliet. Modeling constraints improves software architecture design reasoning. In *Software Architecture, 2009 European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*, pages 253–256, 2009.

[TvV12] A. Tang and H. van Vliet. Design strategy and software design effectiveness. *Software, IEEE*, 29(1):51–55, Jan 2012.

[vdVB13] JanSalvador van der Ven and Jan Bosch. Making the right decision: Supporting architects with design decision data. In *Software Architecture*, volume 7957, pages 176–183. Springer, 2013.

[VFS13] B. Vasilescu, V. Filkov, and A. Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *(SocialCom), 2013*, pages 188–195, Sept 2013.

[vHA11] U. van Heesch and P. Avgeriou. Mature architecting - a survey about the reasoning process of professional architects. In *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 260–269. IEEE, 2011.

[vHAH12] Uwe van Heesch, Paris Avgeriou, and Rich Hilliard. A documentation framework for architecture decisions. *Journal of Systems and Software*, 85(4):795–820, 2012.

[vHJPB+17] U. van Heesch, A. Jansen, H. Pei-Breivold, P. Avgeriou, and C. Manteuffel. Platform design space exploration using architecture decision viewpointsâĂŞa longitudinal study. *Journal of Systems and Software*, 124:56 – 81, 2017.

[vK12] Georg von Krogh. How does social software change knowledge management? toward a strategic research agenda. *The Journal of Strategic Information Systems*, 21:154 – 164, 2012.

[VKZ04] M. Voelter, M. Kircher, and Uwe Zdun. *Remoting Patterns - Foundations of Enterprise, Internet, and Realtime Distributed Object Middleware*. Wiley Series in Software Design Patterns. J. Wiley & Sons, Hoboken, NJ, USA, October 2004.

[WCH18] Shaowei Wang, Tse-Hsun Chen, and Ahmed E. Hassan. Understanding the factors for fast answers in technical q&a websites. *Empirical Software Engineering*, 23(3):1552–1593, Jun 2018.

[WD10] Daya C. Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *J. Inf. Sci.*, 36(3):306–323, June 2010.

[Wen98] Etienne Wenger. *Communities of Practice - Learning, Meaning and Identity*. University Press, Cambridge, 1998.

[WFH11] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[WG14] Rainer Weinreich and Iris Groher. A fresh look at codification approaches for sakm: A systematic literature review. In Paris Avgeriou and Uwe Zdun, editors, *Software Architecture*, volume 8627 of *Lecture Notes in Computer Science*, pages 1–16. Springer International Publishing, 2014.

[Wie09] Roel Wieringa. Design science as nested problem solving. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, DESRIST '09, pages 8:1–8:12, New York, NY, USA, 2009. ACM.

[WKRQ+07] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2007.

[XBL+17] Xin Xia, Lingfeng Bao, David Lo, Pavneet Singh Kochhar, Ahmed E. Hassan, and Zhenchang Xing. What do developers search for on the web? *Empirical Software Engineering*, 22(6):3149–3185, Dec 2017.

[XPK10] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, November 2010.

[YDC+18] Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. Learning to mine aligned code and natural language pairs from stack overflow. In *International Conference on Mining Software Repositories*. ACM, 2018.

[Zim11] O. Zimmermann. Architectural decisions as reusable design assets. *IEEE Software*, 28(1):64–69, Jan 2011.

[ZKL+09] Olaf Zimmermann, Jana Koehler, Frank Leymann, Ronny Polley, and Nelly Schuster. Managing architectural decision models with dependency relations, integrity constraints, and production rules. *Journal of Systems and Software*, 82(8):1249–1267, 2009.

[ZUR+18] Tianyi Zhang, Ganesha Upadhyaya, Anastasia Reinhardt, Hridesh Rajan, and Miryung Kim. Are code examples on an online q&a forum reliable? a study of api misuse on stack overflow. In *ICSE'18: The 40th International Conference on Software Engineering*, May 27-June 3, 2018 2018.

[ZYL+15] Y. Zou, T. Ye, Y. Lu, J. Mylopoulos, and L. Zhang. Learning to rank for question-oriented software text retrieval (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 1–11, Nov 2015.

Eidesstattliche Versicherung

Declaration of Oath

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrifft selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I hereby declare, on oath, that I have written the present dissertation on my own and have not used other then acknowledged resources and aids.

Hamburg, den Unterschrift _____