PERSISTENT IDENTIFIERS FOR EARTH SCIENCE DATA MANAGEMENT

Dissertation

with the aim of achieving a doctoral degree
at the Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics of Universität Hamburg

submitted by

Tobias Weigel

2015 in Hamburg

doi:10.2312/WDCC/tr_d_1

Day of oral defense:

October 19, 2015

The following evaluators recommend the admission of the dissertation:

Prof. Dr. Thomas Ludwig
Prof. Dr.-Ing. Norbert Ritter

ABSTRACT

The advent of widespread developments colloquially subsumed under the notion of data-intensive science poses challenges for data management and end-user applications. The observed increase in volume, variety and number of data objects requires large data infrastructures used for research data management today to further automate their operating workflows. Not only in the Earth sciences, data infrastructures typically rely on distributed middleware services, which must become more scalable and provide more trustworthy and precise information. A widely discussed approach for addressing these challenges is to employ persistent identifiers, traditionally used in the library sciences and scholarly publishing. Such identifiers give globally unique names to digital objects, making them referenceable and possibly accessible independent from their actual storage location. They can also be used by both human and machine agents to retrieve essential state information about the objects.

However, the concept of persistent identification has so far not evolved enough to adequately address data infrastructure challenges. Therefore, this thesis presents a conceptual framework for understanding persistent identifiers in this new context, facilitating solutions that unify access to state information and increase interoperability between distributed identifier systems. Based on a formal model, the conceptual framework defines distinct classification criteria that clarify the differences between identifier usage scenarios and can shape suitable policies of identifier providers. To facilitate interoperability and support scenarios geared towards machine agents, the framework further advocates the use of types to structure state information and to construct digital object collections with unified operations. Existing solutions are shown to partially match the conceptual framework or be adequately extendable, and exemplary Earth science data management usage scenarios can be enabled through its mechanisms. The framework contributes to ongoing international efforts to establish a coherent digital object infrastructure driven by practical needs. In the context of Linked Data, the framework can provide a foundational unification layer and foster the adoption of persistent identifiers for web-based applications.

KURZFASSUNG

Im Rahmen von Entwicklungen unter dem Schlagwort der datenintensiven Wissenschaften ergeben sich neue Herausforderungen an die Verwaltung von Forschungsdaten. Die beobachtete Zunahme an Volumen, Vielfalt und Anzahl von Datenobjekten verlangt von den heutzutage verwendeten Dateninfrastrukturen, ihre Arbeitsabläufe stärker zu automatisieren. Nicht nur in den Geowissenschaften setzen Dateninfrastrukturen typischerweise auf verteilte Dienste, welche skalierbarer werden müssen und dem Nutzer zuverlässigere und präzisere Informationen bereitstellen müssen. Ein vielfach diskutierter Lösungsansatz hierfür beruht auf der Verwendung von persistenten Identifikatoren, welche als global eindeutige Namen für digitale Objekte dienen und für ihre dauerhafte Referenzierbarkeit und potentielle Zugreifbarkeit unabhängig vom tatsächlichen Speicherort sorgen. Darüber hinaus können Nutzer und Software-Agenten gleichermaßen essentielle Zustandsinformationen über Objekte abrufen.

Das Konzept persistenter Identifikatoren ist jedoch bisher nicht umfassend genug gewesen, um den Anforderungen von Dateninfrastrukturen angemessen zu begegnen. Diese Arbeit stellt daher ein konzeptionelles Rahmenwerk vor, welches Identifikatoren in diesem neuen Einsatzgebiet definiert sowie Lösungsansätze unterstützt, die den Zugriff auf Zustandsinformationen vereinheitlichen und zur Interoperabilität zwischen Identifikationssystemen beitragen. Ausgehend von einem formalen Modell werden Klassifikationskriterien entwickelt, welche die Unterschiede zwischen Einsatzszenarien von Identifikatoren verdeutlichen und zur Entwicklung von angemessenen Richtlinien für Identifikationssysteme beitragen können. Um Interoperabilität zu fördern und Einsatzszenarien für Software-Agenten zu unterstützen setzt das Rahmenwerk auf die Typisierung von Zustandsinformationen und den Aufbau von Objektkollektionen mit einheitlichen Operationen. Bestehende Lösungen passen auf das Rahmenwerk mindestens teilweise oder lassen sich entsprechend erweitern, und die vorgestellten Mechanismen erlauben die Umsetzung beispielhafter Anwendungsfälle in der geowissenschaftlichen Datenverwaltung. Das Rahmenwerk trägt zu fortschreitenden internationalen Bemühungen um eine einheitliche Infrastruktur für digitale Objekte bei, die in der Praxis umgesetzt wird. Im Zusammenhang mit Linked Data kann das Rahmenwerk ferner zu einer stabilen Basisschicht beitragen und den Einsatz von persistenten Identifikatoren in Webanwendungen erleichtern.

ACKNOWLEDGMENTS

Creating this thesis would not have been possible without the critical support of a number of people. First, I would like to thank Prof. Thomas Ludwig for supervising me, for giving me counsel whenever needed and for setting essential pointers in unexplored terrain. I am also grateful for the support I received from Michael Lautenschlager, who provided a challenging topic and always kept me involved. I kindly thank Stephan Kindermann for giving profound critical feedback throughout the various development phases of this thesis. I would also like to thank Joachim Biercamp for his support early on.

My gratitude also goes to those with whom I have had extensive contact with over the past three years, especially within the context of the Research Data Alliance. Particularly worth mentioning are Peter Wittenburg, Larry Lannom and Mark Parsons, whose insight and unbroken enthusiasm encouraged me to follow yet unpaved pathways. I would also like to thank Tim DiLauro for sharing the challenges of a guinea pig working group, for his witty humor and his ability to bring a spirit of lightheartedness into potentially stressful situations (including apple pie!), and Tom Zastrow for providing feedback on early chapter drafts and sharing a highly rememberable soccer game.

Last, but not least, I am grateful for the both productive and enjoyable working conditions at DKRZ. My colleagues have always been open for in-depth discussions and shared lunches. Thank you!

CONTENTS

1 INTRODUCTION 1
1.1 Problem statement . . . 3
1.2 Research questions . . . 4
1.3 Methodology . . . 5
1.4 Contribution and relevance . . . 6
2 STATE OF THE ART 9
2.1 Digital preservation . . . 9
2.2 The data life cycle . . . 11
2.2.1 The Digital Curation Center model . . . 12
2.2.2 The curation continuum model . . . 14
2.3 E-science and data infrastructues . . . 15
2.4 Digital objects . . . 16
2.4.1 Identifiers and locators . . . 19
2.4.2 Towards persistent identifiers . . . 21
2.4.3 Persistent identifiers for data . . . 23
2.4.4 The relationship between technology and policies 24
2.5 Conclusions . . . 25
3 USE CASES 29
3.1 The Earth System Grid Federation . . . 29
3.2 Referencing preliminary data . . . 32
3.3 Access to specific versions of a dataset . . . 32
3.4 Referencing custom data slices . . . 33
3.5 Provenance tracing . . . 34
3.6 Discussion . . . 35
4 CONCEPTUAL FRAMEWORK 37
4.1 Formal model . . . 37
4.1.1 Describing identification . . . 38
4.1.2 Describing PID records . . . 39
4.1.3 Further aspects . . . 41
4.2 Persistency layer . . . 42
4.2.1 Fundamental criteria . . . 43
4.2.2 PID system classes . . . 48
4.2.3 Defining the Persistent Entity . . . 49
4.2.4 Abstract data type definitions . . . 52
4.3 Typing layer . . . 54
4.3.1 Typing of identified objects . . . 54
4.3.2 Typing of PID record entries . . . 55
4.3.3 The type registry and type governance . . . 62
4.4 Collection layer . . . 64
4.4.1 Definitions and terminology . . . 65
4.4.2 The collection process . . . 66
4.4.3 Fundamental structural criteria for collections . 68

4.4.4 Common operations on collections 70
4.5 Discussion 70
5 IMPLEMENTATION CONCEPTS 75
5.1 Persistency layer 75
5.1.1 Review of individual PID systems 76
5.1.2 Scalability aspects 79
5.2 Typing layer 79
5.2.1 Type registry 80
5.2.2 Typing layer service 82
5.3 Collection layer 84
5.4 Discussion 87
6 ANALYSIS 89
6.1 Referencing preliminary data 89
6.2 Access to specific versions of a dataset 91
6.3 Referencing custom data slices 92
6.4 Provenance tracing 93
6.5 Discussion 95
7 RELATED WORK 99
7.1 Linked Data 99
7.2 The Entity Name System 101
7.3 Approaches for providing actionable collections 103
7.4 Research Objects 104
7.5 Discussion 105
8 CONCLUSIONS 107
8.1 Future work 108

BIBLIOGRAPHY 111

LIST OF FIGURES

Figure 1	The PID layer cake	7
Figure 2	The Digital Curation Center's data life-cycle model	12
Figure 3	ESGF website: Faceted search interface	30
Figure 4	ESGF website: Dataset with individual files	31
Figure 5	Sets and relations in the formal model	40
Figure 6	The PID redirection layer	44
Figure 7	Architecture of the typing layer	80
Figure 8	Cycle of referenceable data	90
Figure 9	Process flowchart for versioning and provenance	94
Figure 10	Versioning, provenance and organizational hierarchy	95

LIST OF TABLES

Table 1	Type registry data model	60
Table 2	Evaluation of PID systems with fundamental criteria	76
Table 3	Property types for collections	85
Table 4	Exemplary property types for the use cases	98

ACRONYMS

ADT Abstract Data Type

API Application Programming Interface

ARK Archival Resource Key [52]

CMIP Coupled Model Intercomparison Project

CNRI Corporation for National Research Initiatives

DNS Domain Name System

DOI Digital Object Identifier [73]

ESGF Earth System Grid Federation

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

IPCC Intergovernmental Panel on Climate Change

iRODS Integrated Rule-Oriented Data System [76]

netCDF Network Common Data Form, a file format and set of service libraries widely used in the Earth sciences

OAIS Open Archival Information System [70]

OCLC Online Computer Library Center

PID Persistent Identifier

PURL Persistent URL

RDA Research Data Alliance

RDF Resource Description Framework

REST REpresentational State Transfer [29]

URI Uniform Resource Identifier [12]

URL Uniform Resource Locator [12]

URN Uniform Resource Name [12]

W3C World Wide Web Consortium

1 INTRODUCTION

In October 2010, the European Commission's *High Level Expert Group on Scientific Data* published a seminal report entitled "Riding the wave: How Europe can gain from the rising tide of scientific data" [79]. In this report, the experts summarize the current state of the art of European activities regarding large-scale scientific data management and identify the upcoming and unaddressed challenges for the European scientific community. The report advocates the development of a "Collaborative Data Infrastructure", the further construction of scientific e-infrastructures and the recognition of data as primary research output.

The 2010 report clearly identifies the necessary techniques and requirements for scientific e-infrastructures. Among other things, the ideal wish list includes persistent identifiers as a critical base-layer component: "Persistent identification, allowing data centers to register a huge amount of markers to track the origins and characteristics of the information" [79, p. 20]. In the summary of challenges for e-infrastructures [79, p. 22], several related items stand out as well:

COLLECTION: "How can we make sure that data are collected together with the information necessary to reuse them?"

TRUST: "How can we make informed judgements about whether certain data are authentic and can be trusted? [...]"

INTEROPERABILITY: "How can we implement interoperability within disciplines and move to an overarching multi-disciplinary way of understanding and using data? [...] How can automated tools find the information needed to tackle unfamiliar data?"

In the time after the report was published, several related initiatives emerged that take up the report's themes. In 2012, the need for international collaboration on addressing the practical issues in research data management led to the foundation of the Research Data Alliance (RDA), a global initiative that aims to enable data interoperability across boundaries between countries, disciplines, producers and consumers of data [91]. Persistent identifiers (PIDs), roughly defined as names for objects that are stable throughout changes of object location or ownership and are usually globally resolvable, were a central topic among the first set of working groups at the official launch meeting in Göteborg in March 2013, directly reflected for example in the theme of the working group on PID

Information Types[1]. The European Commission also funded the European Data Infrastructure (EUDAT) project in 2011 and tasked it with the development of the Collaborative Data Infrastructure envisioned in the report.

The history of persistent identification per se goes back much further. In the mid of the 1990s, memory institutions such as the national libraries saw the need to expand their mission of preserving cultural artifacts to the contents of the emerging World Wide Web. It was quickly discovered that the dynamic, ungoverned and decentralized nature of the web poses a serious challenge for preservation tasks, not least due to the unstable nature of its Uniform Resource Locators (URLs). A concept for identification and retrieval of digital objects was sought that could meet common preservation requirements. In this setting, the idea of persistent identifiers was born, and several approaches emerged. Among these contemporary *PID systems* or *PID infrastructures*, the most popular ones that are not bound to a specific scientific discipline include Uniform Resource Names (URNs) [63, 71], Digital Object Identifiers® (DOIs)[2] [71, 73] via the DOI System®, Handles® via the Handle System®[3] [45], Persistent URLs (PURLs)[4] and Archival Resource Keys (ARKs) [52]. Although first used for digital articles, it also became clear that the general concept could be applied to data objects as well. With the rising recognition of data as valuable research output, the concept of data citation was promoted, and after a year-long process, the DataCite organization[5] was established in 2009 to offer a formal data citation service based on DOIs. Moreover, initiatives emerge that target persistent identification of entities other than digital articles and data, for example personal identifiers for individual researchers (ORCID)[6] or physical samples (International Geo Sample Number, IGSN[7]).

A prime example for an e-science infrastructure in the Earth system sciences is the international Earth System Grid Federation (ESGF). Originally emerging from a series of US-funded projects, the earlier Earth System Grid (ESG) finally transformed into ESGF as a global open source initiative with partners and funding from the United States, the European Union, Australia and others. Today, ESGF offers a globally distributed infrastructure dedicated to serving the global Earth system science research community. The largest collaborative undertaking in Earth system modeling and an important scientific quality control mechanism are the phases of the Coupled Model Intercomparison Project (CMIP), coordinated by the Working Group

1 The RDA Working Group on PID Information Types was co-chaired by Tobias Weigel and Timothy DiLauro.
2 `http://www.doi.org`, last checked Feb. 27, 2015
3 `http://www.handle.net`, last checked Feb. 27, 2015
4 `http://purl.org`, last checked Feb. 27, 2015
5 `http://www.datacite.org`, last checked Feb. 27, 2015
6 `http://www.orcid.org`, last checked Feb. 27, 2015
7 `http://www.igsn.org`, last checked Feb. 27, 2015

on Coupled Modeling (WGCM) of the World Climate Research Programme (WCRP)[8]. Since 1995, CMIP coordinates a set of standardized experiments performed by multiple international modeling groups every couple of years. Because the amount of produced data has increased exponentially, no single institution is able to serve the complete data output anymore today, so that the results of the fifth phase of the project (CMIP5) [88] have been distributed through ESGF as a community effort for the first time. CMIP5 data finally also formed the basis for the Fifth Assessment Report (AR5) of the Intergovernmental Panel on Climate Change (IPCC)[9] published in 2014. Planning for the next project phase (CMIP6) has already begun [61] and it is agreed that the ESGF e-infrastructure will again serve as the distribution point for model data.

The ESGF infrastructure also faces developments that are evident in all sciences and colloquially set under the umbrella term of *data-intensive science* [38]: As it becomes easier to produce data, either through observation or computational processes [1], the challenges shift towards evaluating and managing data. The climate sciences are no exception, and although techniques including data compression can alleviate some of the data volume [51], the pressure of automating data distribution processes increases and further solutions for data management must be found.

1.1 PROBLEM STATEMENT

All CMIP5 data received DataCite DOIs, however the process was less than optimal and scalability issues are already foreseen for the CMIP6 process. From the users' perspective, the process was far too slow and data could not be cited properly when the first documents had to be submitted for recognition in the IPCC working groups. At the infrastructure side, replication and version management suffered from the lack of a flexible and unified identification mechanism and accompanying policies that aim to ensure timely and safe replication and accurate version tracking. A coordinated application of PIDs and services integrating identifiers, data and metadata may be a way out, however the conceptual details are unclear because there are no proper models for managing data through PIDs, including questions of what to preserve and how to deal with inevitable changes of data objects and ultimately object removal. At the same time, the EUDAT infrastructure makes extensive use of PIDs at its middleware layers, but the solutions are still insular, tied to a specific implementation and infrastructure design and not interoperable across PID systems. A general framework for use of PIDs in e-infrastructures is sought

8 `http://www.wcrp-climate.org`, last checked Feb. 27, 2015

9 `http://www.ipcc.ch`, last checked Feb. 27, 2015

that takes into account the requirements of users, infrastructure providers and preservation institutions.

These issues also hint at a general development at a larger scale. The adoption of web-based solutions continues throughout society and also affects the field of scientific data management. Rather than talking about the management of *files* in file systems, discussions start to revolve around mere *digital objects*. These objects are decoupled from actual storage and are moved and replicated without user's notice. This is particularly evident from the user interfaces of modern mobile devices such as smartphones and tablet computers, where the notions of files and directories become increasingly obsolete. In the area of scientific data management, such objects are managed by automated procedures since the very limited amount of human resources cannot stem the tide of data anymore. These digital objects however still carry some elemental properties which are essential for their proper management. The concept of persistent identification and particularly the notion of Digital Objects as described by Kahn and Wilensky [45] lies at the heart of these recent developments.

While the Semantic Web community has developed a wide variety of solutions for metadata exposition on the World Wide Web, including the notion of making explicit machine-interpretable statements about the relations between web resources commonly denoted as *Linked Data*, such approaches have not seen much practical adoption in Earth science e-science infrastructures such as ESGF for internal data interlinking. The potentially wide range of third-party applications thriving on information exposed as Linked Data remains however a factor that can hardly be ignored. Exposing persistent identifiers and associated information about the Digital Objects behind them as Linked Data may bridge the gap between management solutions and external re-use, yet the role of identifiers and their associated information with respect to Linked Data remains unclear today.

1.2 RESEARCH QUESTIONS

The research conducted answers the following major questions:

What is the relationship between identifiers, data objects, identifier records and metadata? What is the role of organizations and their services towards these entities?

The traditional approach for persistent identification puts emphasis on identifier name schemas and the long-term availability of resolvers. Metadata catalogs may be extended to include identifier information, yet there is no common approach to bind metadata, identifiers and identified objects together. Practical scenarios call for the proliferation

of metadata at a more reliable and scalable service level than can be provided by typical scientific metadata catalogs. However, identifier providers are often not in a position that allows them to safeguard the persistency of metadata catalogs along with the objects. They usually cannot sustain the same level of detail in their services as institutional repositories or e-science infrastructures closer to scientific users.

How do we achieve interoperability between identifier systems or providers, particularly regarding the additional information associated with a PID?

A fair number of identifier systems has evolved over the last 20 years with applications in the scholar community. Each of them follows a distinct approach, and there are fine differences in their conceptual design, fitness for particular application scenarios and policies on persistency and the type of objects that should be identified. Some of these differences are poorly understood however, such as when it is appropriate to assign multiple identifiers, how to assure persistency and how to incorporate relevant metadata.

How can we use identifiers in automated e-science workflows in a scalable and efficient way? How do we deal with increasing numbers of identifiers without compromising their inherent quality? How can we encode and use relationships between large numbers of objects under the constraint that identifiers may only be accessible on an individual basis?

Large data infrastructures are interested in persistent identification of their managed objects to accommodate user needs such as data citation and to reap the benefits of decoupling identification from storage location, particularly if objects are moved or replicated across repositories. However, traditional approaches for assigning and managing persistent identifiers rely on workflows designed with human interaction in mind. Data infrastructures crucially depend on actionability of their process steps through machine agents, yet identifier systems currently fail to address these use cases, particularly in view of scalability for possibly millions of data objects and a large variety of object types from disparate scientific communities. Management at such a scale requires efficient bulk operations and treating data objects with respect to their context and associated entities.

1.3 METHODOLOGY

There are a lot of practical use cases from both Earth sciences and other disciplines that can potentially be enabled through the use of PIDs. Chapter 3 presents an exemplary selection of such use cases which highlights recurring constraints a PID-based solution

has to meet. Together with a careful analysis of the history and existing work in the area of PIDs and scientific data management, this leads to the formulation of a layered conceptual framework for persistent identification of data. The framework can be implemented as a PID service architecture with exchangeable components which is described in chapter 4. Chapters 5 and 6 review the conceptual framework against the motivating use cases to demonstrate how its mechanisms can be applied. Possible benefits and disadvantages compared to alternative solutions are discussed in chapter 7. Chapter 8 closes with some final overarching thoughts and points out areas of future work.

1.4 CONTRIBUTION AND RELEVANCE

This thesis presents a foundational model for distinguishing identifiers, their target objects and associated information and provide clear definitions that can help the actors with defining their roles more adequately in possible future scenarios. If the responsibilities for different aspects of identification and associated services are clearly defined, different actors can achieve a level of independence with respect to their core responsibilities, while the solution as a whole is still able to satisfy core requirements for persistent identification. This includes information associated with PIDs and available at a primary level of preservation independent from the availability of target objects.

Although the final decision what metadata deserve to be more closely associated with identifiers can only be adequately addressed from a user perspective, a set of distinct stages of information can be identified and help in making such decisions. The framework is based on the themes of organizing information and services in an architecture with abstraction layers dedicated to fundamental requirements and persistency, typing and iterable collections (see figure 1). All layers rely on individual PIDs as binding elements and the main mode of access. The framework can enable interoperability between PID providers, including interoperability of essential PID-related metadata typically stored in PID records.

From the representative use cases it is evident that the management of object identifiers relies on workflows that can be modeled through applications of fundamental abstract data types such as lists, sets and graphs. These concepts have been known in computer science for a long time, and they are sufficiently generic to be applied to the processes in data infrastructures. The central idea is to enable machine-actionability of identifiers and understand the persistent entities which they refer to as instantiations of a generic abstract data type with a set of clearly defined methods enabled across identifier providers and independent from the object types. A practical aspect

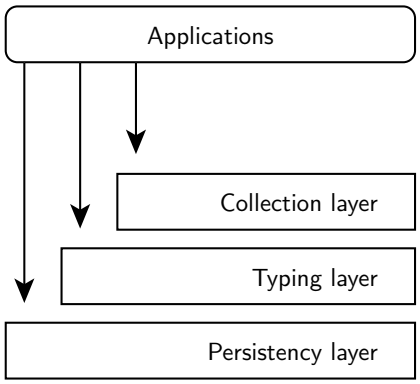

Figure 1: Applications may enter the architectural layer cake at any layer as appropriate for the specific use case.

of the framework is thus to enable actionable identifier collections where identifiers are treated as first-class entities while black-boxing the associated metadata and target objects.

Although the framework can possibly be transferred to other disciplines, the motivating use cases originate in application scenarios from a discipline-specific data infrastructure (ESGF). The use cases put special emphasis on processes concerning data replication, versioning and provenance tracing. Basing core processes of data infrastructures on identifiers also requires a set of tools for analyzing identifiers, the structures built with them and associated metadata. Chapter 6 includes concepts for implementing such tools based on the developed conceptual framework.

Aside from the use of identifiers in current e-infrastructures such as EUDAT and ESGF, there are a lot of recent international initiatives related to persistent identification. For example, the ODIN project aims at building interconnecting services between DataCite and ORCID. The administrative responsibility of the Handle System is also in a transition process to being handed over to the newly founded Digital Object Numbering Authority (DONA) under the auspices of the International Telecommunication Union (ITU). A generalized framework on persistent identification of digital objects that takes into account the needs of e-infrastructures on automated data and metadata management may prove to be valuable to these initiatives.

2

STATE OF THE ART

The dissemination, archival and identification of research output requires a broad understanding of the processes, technologies and conceptual models involved. This chapter provides a comprehensive literature review to determine the essential concepts of digital preservation, the data life cycle, persistent identifiers and digital objects, and illustrates how the defining aspects of persistent identifiers are shaped by practical concerns. Special emphasis is put on the historical development of the concept of persistent identification, which is tightly interwoven with the history of Uniform Resource Identifiers and the World Wide Web.

2.1 DIGITAL PRESERVATION

With the advent of information technology, institutions charged with the archiving of research output recognized that digital objects must be preserved just like physical objects, spawning activities under the umbrella of *digital preservation*. A definition for the activity of digital preservation is for example given in the Open Archival Information System (OAIS) reference model [70]:

> "Long Term Preservation: The act of maintaining information, in a correct and Independently Understandable form, over the Long Term."

This definition emphasizes a key issue of preservation: information remains useful only if it can be understood independent from the original issuers or authors whose full interpretative knowledge is expected to become unavailable over time. The exact time span for preservation ("long term") is a matter of policy and shaped by a number of practical and idealistic considerations. Typical time spans given in the literature vary from 10 years [24] to even potentially indefinite time [70]. Digital preservation has also been described as "communication with the future" by Moore [64, p. 64]. This is reflected in another definition for digital preservation provided by Jantz and Giarlo [44]. Based on a report by the Online Computer Library Center (OCLC) [92], Jantz and Giarlo define digital preservation as: "the managed activities necessary: 1) For the long term maintenance of a byte stream (including metadata) sufficient to reproduce a suitable facsimile of the original document and 2) For the continued accessibility of the document contents through time and changing technology." [44, p. 136]

The world wide web has increased production, distribution and re-use of digital objects, which presents a particular challenge to the preservation efforts, and in particular the problem of broken web links has been detected early on [59]. In 1996, a US task force report (see [75]) was published which set the initial conditions for the developments in digital preservation ever since. It recognized that there are several levels of sophistication for digital preservation, with the lowest level simply focusing on preserving the bit-stream. As the report states, the pure technical view of a preserved bit-stream is however often not sufficient for ensuring future usefulness. Such a view neglects encoding formats and other structural dependencies on changing technology — there is additional context associated with a digital object that needs to be preserved as well, which resonates with the notion of an "Independently Understandable form" as stated in the OAIS definition above. At a higher level of abstraction, digital archives therefore aim to preserve "content in terms of the knowledge or ideas the object contains" [75, p. 19], which requires migration strategies to cope with technological advancement.

Digital preservation thus aims to preserve more than just the bit-stream. The additional information for a digital object is often described through its metadata, though as Giaretta [32] remarks, the term "metadata" is highly overloaded and issues such as what "types" of metadata are relevant and "how much" metadata is required are unclear. A well-accepted model for digital preservation that includes metadata aspects is the aforementioned OAIS Reference Model, which was endorsed as an ISO standard in 2003. The OAIS reference model defines a group of metadata types (the "preservation description information"): "The information which is necessary for adequate preservation of the Content Information and which can be categorized as Provenance, Reference, Fixity, Context and Access Rights Information." [70] These concepts are similar to the features already mentioned in the 1996 task force report: "In the digital environment, the features that determine information integrity and deserve special attention for archival purposes include the following: content, fixity, reference, provenance, and context." [75, p. 10]

Reducing digital preservation to purely technical aspects is however a misconception. As explained in a more recent report from 2013 [27], digital preservation always relies on management and organizational policies; technology in itself cannot ensure preservation. This is also reflected by the notion of a "Trusted Digital Repository"[1] as defined by the OCLC, which requires conformance with the OAIS reference model but also several policy aspects including reliable administration and financial and organizational sustainability [92].

1 also see `http://www.trusteddigitalrepository.eu`, last checked Feb. 27, 2015

2.2 THE DATA LIFE CYCLE

Curation processes required for research data go far beyond the scope of digital preservation [3]. This extended view is also known under the term *digital curation* [103, 7], which spans the entire life cycle of digital information. But what exactly is this life cycle? To answer this question, it is necessary to first examine the overarching concept of digital curation and its history.

Beagrie [7] explains the history of the term and concept of digital curation. Digital curation stems from the earlier concept of digital preservation, but transcends it as an activity that provides some form of added value, such as building library or museum-like collections, building long-term knowledge about the objects and providing context for further use. Digital curation also tries to involve data creators and researchers in activities that are neglected in the more limited scope of preservation and archiving. In this respect, Beagrie explains how preservation was commonly perceived as an "end-of-project activity" that may lead to fragmented preservation and "data mortuaries" [7, p. 5]. To extend the scope of digital curation beyond these limitations, Beagrie and others mentioned by him (such as the JISC) argue for a more holistic view on the data life cycle, where preservation is just one activity among others. This culminates in a definition of digital curation as "the actions needed to maintain digital research data and other digital materials over their entire life-cycle and over time for current and future generations of users"[2] [7, p. 4].

Yakel [103] provides an insightful review of major activities and reports published around 2007 when there was a significant research interest accumulating in the general area. She identifies five characteristic common concept or activity areas for digital curation, all of which target the long term [103, p. 338]:

1. Life cycle/continuum management of the materials perhaps even reaching back to the creation of the record keeping system.
2. Active involvement over time of both the records creators and potentially digital curators.
3. Appraisal and selection of materials.
4. Development and provision of access.
5. Ensuring preservation (usability and accessibility) of the objects.

2 Primary references given by Beagrie include the JISC at `http://www.jisc.ac.uk/uploaded_documents/6-03%20Circular.doc` (unavailable) and the DCC at `http://dev.dcc.rl.ac.uk/twiki/bin/view/Main/DCCApproachToCuration` (unavailable); last checked Feb. 27, 2015

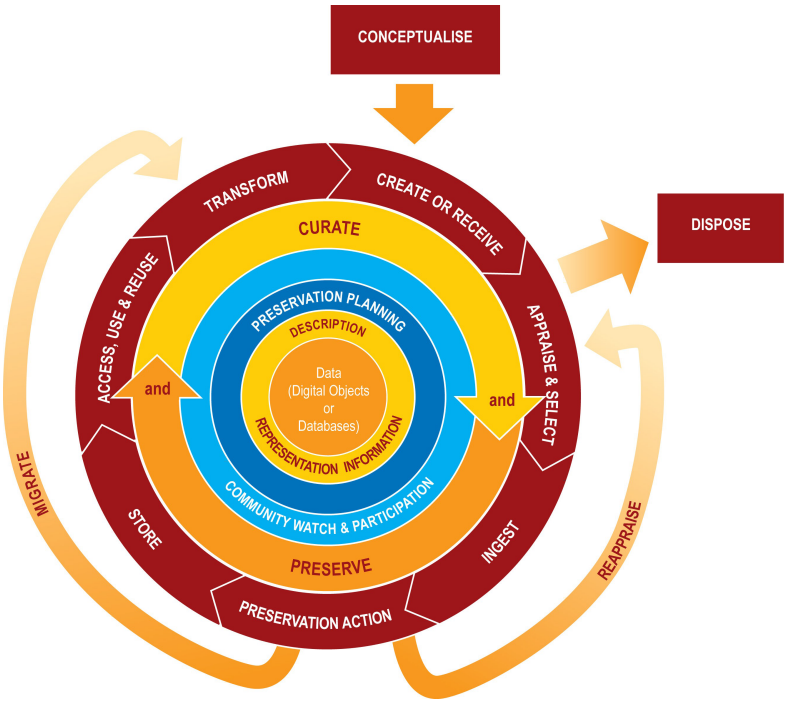

Figure 2: Graphical representation of the Digital Curation Center's data life cycle model (taken from [40, p. 136]).

Yakel also remarks how (at the time of publishing) digital curation was becoming an umbrella concept for the interrelated areas of digital preservation, data curation, digital asset management and electronic records management.

Various models exist in the literature for describing the general data life cycle. Typically, a data life cycle model will define specific stages through which data proceeds over time.

2.2.1 *The Digital Curation Center model*

The curation life cycle model of the Digital Curation Centre (DCC), presented by Higgins [40] (also see figure 2), is widely used and referenced model. As is also clearly stated on the DCC website, the model is idealized, and "[i]n reality, users of the model may enter at any stage of the life cycle depending on their current area of need."[3] Nonetheless, the stages included in the model are useful enough to cover a broad range of disciplines.

The entities the DCC Model aims to curate are separated into two rough categories: Digital Objects and databases [40, p. 137]. The digital objects are further subdivided into simple objects and more complex ones that are created by combining other digital objects.

3 `http://www.dcc.ac.uk/resources/curation-lifecycle-model`, last checked Feb. 27, 2015

Databases are characterized as structured collections of records or data.

At the highest organizational level, the model introduces four overarching so-called "full life cycle actions" and a number of more fine-granular sequential actions. The overarching actions span the life cycle, while the sequential actions are more low-level and address distinct steps or stages. The collective of sequential actions therefore cannot exist independently from the larger overarching concerns. Instead, the sequential stages are rather implementations of the overall goals expressed by the overarching concepts. These full life cycle actions include: to describe and represent information, to establish management and administration plans for preservation, to maintain a community watch and participation strategy and to execute curation and preservation actions according to the plans. More detailed descriptions for the actions can be found in [40].

The individual sequential actions listed as the outermost rim in figure 2 together form a collective characterization of the stages data passes through during its life-time. As Higgins explains, the actions are taken in order, but not all of them may be applicable to every domain. Instead, the model should be used to arrange actions in a proper order, identify gaps and decide if particular actions can be omitted or are inapplicable for a specific domain. The set of sequential actions is quite exhaustive; the interested reader can find more in-depth descriptions of the individual steps in [40].

In addition to the sequential actions, the model also encompasses a few "occasional actions". In contrast to the sequential actions, occasional actions express alternate flows that are only taken if certain conditions are met. For example, the "dispose" action is particularly important for triage of data — not all data will be selected for further curation, and consequently the dispose action offers a proper exit point for such data. This means in particular that even the disposal of data must be well managed to assure data quality. Other occasional actions include to reappraise data if they fail validation before entering storage and to migrate data to new formats, after which the data must be subjected to a new cycle to be allowed into the preservation process in a controlled manner.

One observation to make at this point is that the DCC model does not specifically include a possibility to share data and enable reuse before preservation actions are taken that imply a goal of long-term archival. Such scenarios are however quite common in everyday science workflows and e-science in particular; data are re-used after some controlled metadata gathering and QA processes have taken place but before they enter long-term archival stages. A particular example for this is the ESGF workflow (also see chapter 3), where quality-controlled data products enter a data infrastructure geared towards distribution and re-use that however does not aim to

provide long-term archival. Such a scenario effectively replaces the preservation and storage stages with a much less controlled and less accountable environment, though this does not exclude such stages from taking place at a later point in time. In contrast, the DCC model only offers re-use of data objects that have been preserved. These different viewpoints and realizations are not outright conflicting, but should rather be seen as the possible ends of a spectrum, where each discipline, infrastructure and strategic approach may be located in.

As indicated by Higgins [40], the DCC model may be modified by particular domains to accommodate specific needs. For climate community processes, Lautenschlager [54] states that the most relevant sequential actions are creation, processing, archival and re-use. The archival stage may also be omitted if long-term archival is not desired or done later (also see the ESGF use cases in chapter 3). Another relevant contribution is presented in [27]. Here, the cycle model is a simplified descendant of the DCC model. This model includes only six sequential actions: planning/creation, selection, ingest, storage/infrastructure, preservation, access/usage. The set of overarching concerns is however more detailed and also includes for example legal and ethical concerns, financial concerns (including long-term cost models) and issues of unique identification. This latter point is particularly significant, as it calls for processes and policies that must be put in place to include PIDs in a quality-assured manner across the life cycle stages.

2.2.2 *The curation continuum model*

Treloar and Harboe-Ree [90] describe how curated data have a broad range of management features. They identify several distinct continua or dimensions: metadata, item count and size, item dynamicity, responsibility for curation divided between researchers and organizations, degree of preservation, wideness of access, exposure or searchability. Each curated object will be located somewhere in this multi-dimensional space, but the particular location may change as the object passes the stages of its life cycle.

Given the multi-dimensional space, Treloar and Harboe-Ree proceed to identify three distinct domains and boundaries between them, ranging from the private research domain over a shared research domain towards a public domain. Data passes through these domains in general from private to public over its life-time. Transitions between the domains often mark crucial phases in the life cycle of a data object which for example involves shifts in requirements or in the responsibility for curation. The domains are defined by their respective coverage on each of the dimensions, e.g. whether there is more metadata available or less or whether objects tend to be numerous and voluminous or small, static and distinct. As

Treloar and Harboe-Ree remark, their tripartite arrangement is just one possible example for defining such a set of domains (although one that has apparently been picked up by the larger community in a favorable way).

The practical application of the model finally is to map existing solutions to the domains. Each of the specific domains has its own preferred generic class of repositories and supporting systems, though Treloar et al. emphasize that there is no strict one-to-one relationship towards actual systems. One operative repository system may for example be used for two or even all domains, though such setups are unlikely to work out in practice.

The Radieschen project [47] proposed to include a fourth domain, described as the "access domain", because in their view, the model with three domains does not work well for cross-disciplinary workflows and cases where there is non-public long-term archival. They also differ from Treloar and Harboe-Ree in their definition of the other three domains since they relocate the boundaries between public and private aspects. Unfortunately, they do not provide a more exhaustive definition of these domains in terms of how it covers the continua identified by Treloar and Harboe-Ree.

2.3 e-science and data infrastructues

According to Hey and Trefethen [39], e-science can be characterized as the change in methods and tools employed in some major scientific disciplines that is driven by the deluge of data available from high-performance computing experiments, sensor networks, satellite surveys and so on. These changes in working mode and consequently in the needs of scientific users put development pressure on the general IT development for science, requiring more sophisticated search and cataloging facilities with automated metadata creation, data mining and visualization tools. These various elements of IT-related development have been collated under umbrella terms such as e-science infrastructures or cyberinfrastructures that also build upon the earlier "science grid" developments (see e.g. [9, 30]). As Hey and Trefethen remark, e-science is not a new scientific discipline in its own right; rather, the e-science infrastructure offers fundamental support for scientist to practice faster, better or different forms of research.

Later, in 2010, Gray [38, p. xviii] goes as far as proclaiming e-science to be a new fourth paradigm of science, an extension of the scientific method originally consisting of empirical, theoretical, and computational paradigms. Other terms in use for this latest evolution of the methodological changes and aspects of the area where "IT meets scientists" (as Gray calls it) are data-intensive science, data science and to some extent big data with a focus on science. Gray also describes how data analysis and delivery in particular pose to

be a problem and motivates to move queries to the data as opposed to the former default scenario of moving the data to the scientist's workbench. The challenges are also by far not confined to the physical science domains which have a tradition of producing and analyzing large amounts of data. Social sciences dealing with data face similar challenges [46] with very diverse, complex and heterogeneous data [43].

At earlier life cycle stages, the problem of data delivery has given rise to some extensive e-science infrastructures geared towards data access, cataloged search and initial integrated data analysis and processing capabilities. One popular example from the domain of high-energy physics is the Large Hadron Collider computing grid (LHC grid) [53, 80]. The challenges are not limited to the physical sciences, however, as the example of the Digital Research Infrastructure for the Arts and Humanities (DARIAH) illustrates.

In climate science, an e-science infrastructure with similar goals to the LHC grid was the US-based Earth System Grid [13], later succeeded by the international Earth System Grid Federation (ESGF) [100] which continues to operate and provide services for the CMIP5 and future CMIP6 data. Budich and Hiller [18, p. 1] point out that a particular problem in the Earth system modeling domain not present in other domains is the necessity to provide data to a massively growing community of researchers and laymen due to the interest in climate change and its impact, which goes beyond simply dealing with the increase in data volume and variety.

The quality control process employed in ESGF for the CMIP5 data described by Stockhause et al. [85] illustrates how the challenges of data-intensive science also spread across the stages of the data life cycle. Stockhause et al. describe how the preservation of large amounts of research data is a challenge that requires automation of central processes so that the digital curation of research data continues to scale well in view of limited personnel resources.

2.4 DIGITAL OBJECTS

There are many definitions for digital objects in the literature.

A quite straightforward and general definition is given by the OAIS reference model [70], which defines a digital object as "an object composed of a set of bit sequences." This definition is quite broadly applicable, yet it does not specify more details about the composition of a digital object or its more specific usefulness and interpretability.

In contrast, Thibodeau defines digital objects with much more distinct aspects: "Every digital object is a physical object, a logical object, and a conceptual object, and its properties at each of those levels can be significantly different. A *physical* object is simply an inscription of signs on some physical medium. A *logical* object is an

object that is recognized and processed by software. The *conceptual* object is the object as it is recognized and understood by a person, or in some cases recognized and processed by a computer application capable of executing business transactions." [89, p. 6]

This definition accounts for the complex meaning of a digital object for human and machine actors, and also provides a possible distinction between bit-stream preservation and efforts beyond that.

Thibodeau also states that a logical object may be a composite of other logical objects. If the composite is (physically) broken into parts, i.e. if the individual logical objects are extracted and stored as distinct physical objects, keeping the whole composite intact requires setting links in the logical composite object to provide references to its parts. "The way they are stored is irrelevant at the logical level" [89, p. 8], which is an expression of layering and decoupling the physical storage from the logical aspect. Thibodeau concludes that such a scenario prerequisites that "every logical object have its own persistent identifier, and that the location or locations where each object is stored be specified." [89, p. 8]

An important question is what distinguishes a logical object from a conceptual object. Thibodeau states: "The conceptual object is the object we deal with in the real world: it is an entity we would recognize as a meaningful unit of information ..." [89, p. 8]. "In the digital realm, a conceptual object may also be one recognized by a business application, that is, a computer application that executes business transactions." [89, p. 8] Thibodeau emphasizes that the conceptual structure may differ substantially from the structure of the logical object. Thibodeau also describes how there are relationships from one-to-one up to many-to-many between any two levels of physical, logical and conceptual objects. Thibodeau concludes that "in order to preserve a digital object, we must be able to identify and retrieve all its digital components", which are "the logical and physical objects that are necessary to reconstitute the conceptual object." He also concludes that "digital preservation is not a simple process of preserving physical objects but one of preserving the ability to reproduce the objects", while objects here should mean conceptual objects. [89, p. 12]

Allison et al. [2] compare digital objects with classic physical documents and argues that just as there are means to proof the authenticity of physical documents, so there must be an analogon for digital objects. According to Allison et al., authenticity concerns include prevention of forgery, making transactions (changes, copying, transfer of ownership) on a document verifiable and managing documents through a reliable registry. They point out that a digital object, in contrast to a physical object, cannot be perceived directly by human observers, but must be subjected to a chain of stages and processes to be perceivable, and thus the actual impression is

influenced by changes to this chain in addition to the changes in the digital object's actual bit-stream.

The work by Kahn and Wilensky, commonly referred to through a paper published in 2006 [45], is in fact much older. As stated in the preamble of the 2006 paper, the original text is as old as 1993 to 1995, a time where the concepts for digital preservation of objects available through the Internet were relatively young. The Handle System, part of the Digital Object Architecture envisioned by Kahn, has been operational since 1994.

Kahn and Wilensky define digital objects as tuples consisting of digital material, key-metadata and a *handle* included in the key-metadata, which persistently identifies the digital object. The exact definition is as follows [45, p. 117]:

> "Formally, a digital object is an instance of an abstract data type that has two components, *data* and *key-metadata*. The data is typed, as is described below. The key-metadata includes a *handle*, i.e., an identifier globally unique to the digital object; it may also include other metadata, to be specified."

The definition puts particular focus on the persistent identifier, the *handle*, which is intrinsically tied to the digital material and the overall digital object. Overall, this is a much more technical definition than the definitions reflected above. Kahn and Wilensky do not describe what kind of metadata is associated with the digital material [45, p. 117]:

> "No attempt is made in this paper to delineate how much of the metadata should be included in the key-metadata, other than requiring that it include the mandatory handle."

In the Flexible and Extensible Digital Object and Repository Architecture (FEDORA)[4] [74], based on the Kahn and Wilensky framework, a "DigitalObject" is described as conceptually consisting of two parts: "(1) a structural kernel, which encapsulates content as opaque byte stream packages and, (2) an interface, or behavior, layer that gives contextual meaning to the data in the DigitalObject." [74, p. 42] The outer layer must be understood as providing content dissemination functionality, components which can transform the kernel data to diverse information entities for particular audiences and contexts. FEDORA also defines a number of service requests for whole digital objects, including replication and transfer of objects between repositories.

4 Also known as FEDORA Commons, `http://fedora-commons.org`, last checked Feb. 27, 2015

The Open Archives Initiative Object Reuse and Exchange specifications (OAI-ORE) describe compound information objects as "aggregations of distinct information units that when combined form a logical whole"[5]. The distinct units may be different representations of the same content, but also items that thematically belong together. All internal components and the whole object are identifiable through URIs and machine-interpretable relations between components and also between different information objects.

2.4.1 *Identifiers and locators*

As already seen in some of the conceptualizations revolving around digital objects, the question of how to identify an object is elemental. In some models, such as the Kahn and Wilensky framework, the identifier is integral part of the digital object. To understand how the two issues of digital objects and identification are related, it is important to look at the history of identifiers, and first to understand the issues of identifiers and locators.

The problem of "link rot" or "URL decay" [49, 55, 83, 41, 101][6] did not become evident until the actual creation of the World Wide Web and URLs as the means to locate objects. With URLs also came the distinction between URIs, URLs and URNs. These historic developments must be outlined first to understand the historic context in which early PID systems (Handle, ARK) were created. The concepts of URLs and URNs appeared together at about the same time and complement each other. The first solid distinctions regarding URIs, URLs and URNs can be found in IETF RFC 1630 [10] from 1994, which most particularly specifies URLs as responsible for object access over protocols and URNs as more persistent names than URLs. A more thorough formal definition for URNs appeared a bit later in 1997 in IETF RFC 2141 [63], where URNs are defined as serving as persistent, location-independent resource identifiers.

The 1996 report [75] falls in the historic context of early URL, URI and URN discussions. It states that to ensure information integrity, objects must have a consistent means of reference. The authors see systems for citation as one solution to provide consistent references; such systems stem from the library science area rather than the web science area. URLs are seen rather critical since they are location specific and frequently change as objects are moved. At that point in time, the potential location-independent alternative, namely URNs,

5 `http://www.openarchives.org/ore/documents/CompoundObjects-200705.html`, last checked Feb. 27, 2015

6 As URLs are in the context of this thesis considered to be rather unstable, their usage here is of course a peculiar and slightly ironic topic of concern. Lacking the trustworthiness of proper PID systems, all URLs given in this thesis have been judged individually by the author in terms of how far he trusts the responsible maintainers to ensure their long-term resolvability.

was still a very academic concept not put into wide practical adoption, as is stated in the report as well. The later history shows that in fact URNs have stayed at a rather neglected state of practice ever since and by far did not achieve a comparable level of adoption like URLs (i.e. "http:" URIs, see below) with the notable exception of use in the library and archival community.

The respective roles of URIs, URLs and URNs were clarified later in 1998 with RFC 2396 [11], where URLs and URNs are seen as subclasses of URIs, which are not necessarily disjunct. This finally culminated in a revised view expressed in RFC 3305 [60] from 2002, which acknowledges the practical development of the world wide web and the way in which URLs have been used since their inception. URIs are defined as providing a generic syntax (defined in RFC 2396 [11], which was superseded by RFC 3986[12][7]), part of which are specific URI schemes. A URI can be further classified as a locator, a name, or both [11]. URNs are names, expressions of one of the possible URI schemes: "urn:". A URL (a locator-type URI) is redefined as "a type of URI that identifies a resource via a representation of its primary access mechanism (e.g., its network "location")." A URL can thus be seen as an identifier of a location. An important criterion to note is the primary access mechanism, an example for which is the "http:" scheme of a typical web-URL. The mandatory specification of a primary access mechanism is a main conceptual distinction from the classical view on URNs: URNs do not imply that a resource can be accessed using the URN, because an access mechanism may be unknown or even not existing or because the resource may become nonexistent [11].

As Paskin [73] points out, the URI and URN specifications do not provide implemented infrastructures and accompanying services. While the Domain Name System works well for URL resolution, a similar implemented system for global URN resolution is still missing. The total set of all URNs is subdivided into so-called URN namespaces which do not overlap. Some national libraries have been establishing URN resolution services for the particular namespace of National Bibliography Numbers (NBNs).

A contemporary view on identifiers and locators with particular emphasis on object replicas and application to Earth system science data is given by Duerr et al. [26]. They see identifiers and locators as being defined by the respective requirements of *location independence* and *location invariance*. According to their definitions, a *location independent identifier* identifies every copy of an object, no matter where it is located, and is potentially contained within each object. A *location invariant locator* points to an object, but is not necessarily

7 RFC 3986 [12] provides more elaborate considerations on the meaning and definition of a URI, but does not change or contradict the definitions given in the earlier RFC obsoleted by it [11].

located within it, and also does not point to any arbitrary copy of an object, but to a specific (authoritative) one.

Duerr et al. also require for a location independent identifier to be generated at the time an object is created, to be placed within the object itself and to be referenced within related information about the record such as a metadata record. These are additional requirements, however it is unclear why the location independence implies them. Thus they must stand as separate but nonetheless useful requirements. The same applies to an additional requirement from the geosciences mentioned by Duerr et al. that a location independent identifier must not require a naming authority, motivated by the idea that such an identifier should be available during fieldwork where Internet access is unavailable and that the identifier should be persistent, i.e. not modified after initial assignment.

Kunze [52] puts emphasis on the association (the binding) between an identifier and its target, in this case an information object, which incorporates a record of associated information in addition to the referenced object.

2.4.2 *Towards persistent identifiers*

The problem of decaying URLs and broken links is certainly as old as the world wide web itself and, unsurprisingly, the attempts to overcome this problem have been going on for almost two decades. Within these two decades, various attempts have been put forward, yet the problem remains unsolved at a broader scale. The concept of a *persistent* identifier expresses the wish for an ideal solution where the same name can be used over time to retrieve a resource, no matter where it currently resides or who owns it, and thus stands in contrast to the supposedly fragile URLs.

The term *persistent identifier* (abbreviated as PIDs or, more rarely, PIs) has been around for quite a while. Starting with the DOI system, contemporary literature uses the term for a concept that seems quite clear at a first glance. In detail, there are however important requirements inherent in different views. Most notably, the term identifier, as opposed to a locator, seems to be set – a "persistent locator" is somewhat unheard of. As described further below, *resolvability* is largely implicit in viewpoints that simply talk about "persistent" identifiers.

Perhaps as a symptom of the remaining level of ambiguity in the academic discussion, literature dealing with persistent identifiers frequently tries to disambiguate the concept by putting more precise adjectives in front of the ambiguous noun "identifier". As a result, there are unique, resolvable, persistent, interoperable, or even citable identifiers. Paskin [73] makes an attempt to define these adjectives, which are implicit requirements for identifiers, more precisely:

UNIQUENESS : This can be read in two ways. According to Paskin, it may mean that "one string denotes one and only one entity", which does not necessarily imply that every entity has at most one identifier pointing to it. An equivalent description is that there should be only one namespace. Coyle [21] contrasts this with the notion that within a defined context, uniqueness may in fact mean that "each entity has one and only one identifier". Even for the case where uniqueness refers to a single entity being identified, it is again context dependent what this means; does an ISBN, for example, identify a book edition or every physical copy of it?

RESOLUTION: For a *resolvable* identifier there must be a service that takes the identifier string as input and returns a specific output related to the identified entity.

PERSISTENCE: Once assigned, an identifier denotes the same entity indefinitely.

INTEROPERABILITY: An interoperable identifier can be used in services outside the direct control of the identifier assigner. Assumptions made on assignment must be provided.

Paskin [73] points out that specifications such as URN and URI are just that — specifications, but not implemented infrastructure and accompanying policies. Paskin suggests to employ identifiers that fulfill all four criteria to overcome the problem of decaying locators. The four criteria are at the core of the DOI system design, which Paskin helped to establish.

Other requirements seen in the literature are:

CITABILITY: Duerr et al. require that citable identifiers "need to be broadly accepted and used by journal publishers" [26, p. 143]. Although there are also technical requirements for citability – according to Duerr et al., a citable identifier must be unique, location invariant and point to the current location of the data – the signifying criterion for citability is given by such policies with the publishers. The identified object must also be subjected to digital preservation (e.g. through a long-term archive) [48].

ACTIONABLE: Kunze [52] and Paskin [72] define an *actionable* identifier as one that in the context of a particular infrastructure, e.g. the world wide web, can be interacted with by an end-user in a straightforward way, for example in a web browser, to directly or indirectly gain access to the referenced object. The last mile of this interaction process may be extended through HTTP content negotiation to adequately serve human and machine agents.

Jantz and Giarlo describe the concept of a persistent identifier as follows: "We would like to assign a globally unique name to a

digital object, a name that can be used, in perpetuity, to refer to and retrieve the digital object." [44, p. 140] This can be rephrased as requiring an identifier to be unique, persistent and resolvable according to the definitions given above. It can also imply that the entity is never deleted and that resolvability requires the entity to be returned and not just related output. However, these two things are strictly speaking not concerned with the core mechanisms of identifying an entity, but rather focus on retrieval aspects of digital preservation that go beyond identification. An entity can be deleted and still the persistent identifier remains, since the name of an entity does not disappear when the entity does.

2.4.3 *Persistent identifiers for data*

What are the digital objects PIDs are assigned to? One popular PID system with a long history is the Handle System. Since its conception, it has been proposed and used in various areas, including content from the media industry and the U.S. military. One particularly large community, not just for the Handle System, but for PID systems in general, has been the scientific publishing community. Here, the problem of decaying URLs raised serious concerns when scholarly communication started to use the Internet as a mainstream medium [41]. The DOI Foundation is therefore still the perhaps most prominent customer of the Handle System.

More recently, interest has risen within the scientific community to assign identifiers to objects other than scholarly articles. The first milestone in this development is the DataCite initiative, officially founded in 2009 as the final outcome of collaborative effort going on since the early 2000s [17], where identifiers are assigned to scientific datasets with the explicit goal to make them citable at the same level as articles, the more traditional research product. More recent examples are identifiers for people, academic researchers in particular, represented by activities such as ORCID, or identifiers for physical objects in science such as the International Geo Sample Number (IGSN).

While classic PID systems and literature discussions focus on the notion of identification, redirection and occasionally naming schemes (as stand-ins for actual implementations), there is also a fundamental discussion that revolves around the notion of additional information associated with such identifiers. While the intent to identify and ultimately locate an object over its lifetime and through changes in location or ownership is still present, being able to retrieve additional context information on the object using the same identifier is seen as increasingly useful. This resonates with questions and intentions present at the earlier days as well, and subsequently reflected in the

OAIS metadata types to provide accountability and verifiability of objects.

The implications of such a view are manifold. A view focused on identifier names and schemes for them cannot cover the additional information adequately, since that would require to look more deeply into aspects of how this information is stored and accessed.

On identifier names, Jantz and Giarlo remark that "the PID naming convention should generally be free of technology dependencies, protocols, and local naming conventions" [44, p. 141], because technology and naming conventions change over time. A more detailed analysis of such *intelligent identifiers*, which enable some interpretation outside their registration domain via the identifier name, is given by Paskin [72] with a similar conclusion, which largely advises against semantically rich identifier names.

A particular issue is that of object fixity [75], i.e. providing mechanisms that can validate that an object has not been altered. For valuable resources which are in fact continuously updated, the complete record of changes should be maintained. The report describes this in terms of databases only, however the concept is applicable in general (not every dynamic object today is residing in a database). The report authors also admit that providing such level of detail reliably can be expensive to maintain. The OAIS model includes such fixity and provenance information.

2.4.4 *The relationship between technology and policies*

The solution to overcome the URL decay problem is not primarily of a technical nature. In the case of HTTP URLs for example, the actual mechanisms that can enable stability of the binding over longer time spans have been known for quite some time now: the HTTP specification [28] lists several methods for redirecting look-up requests, each with its own distinct semantics. The PURL system in particular makes use of these mechanisms. Beyond HTTP, there are also various other technical methods, from server configuration options down to the low level mechanisms of the Domain Name System. So if all those technical mechanisms exist, why is the problem still unsolved?

According to Hilse and Kothe [41], URLs break due to two reasons: either documents no longer exist or they do exist, but were relocated to a different path or domain. Hilse and Kothe continue to explain that, from a theoretical perspective, these problems are largely due to inadequate administration of the respective servers. In practice, the problem is more complex particularly due to the domain name being part of the URL. Domain names change because they are bound to trademarks, companies or individual departments, and these changes cannot be prevented easily. Hilse and Kothe therefore conclude that

persistence is a function of organization and administration, not of technology.

So an obvious answer to the question stated above is that establishing the concept of PIDs is not achievable by technical means only. Rather, social and legal policies, quality-controlled management processes, contingency strategies and not only a general awareness, but somewhat collaborative responsibility for the issue are needed. Von der Hude [42] lists a reliable resolution service, a commitment to long-term archival and usage policies as the main requirements for URN persistence in particular. Nicholas et al. [69] point out that dependency on particular technologies may actually turn out to be a hindrance for persistence, since technological progress must be dealt with as well.

One strategy to at least ease the costs of maintaining persistence is to provide the technical means to support such policies. Plain URLs in general do not bear information on the level of commitment regarding the persistence or time span of validity. Kunze [52] therefore argues for introducing a commitment statement and presents an insight that clarifies a potential socioeconomic factor. In his view, a simple indirection layer is an inferior solution compared to a commitment statement, because this puts an inversion to the value of the things identified: Although a bad DNS entry may be fixed at high priority, an individual object's identifier is regarded with much less value and thus remains unfixed. If a provider is instead obliged to provide a commitment statement, the user can transparently see what the preservation policy of the issuer is and in the best case expect the identifier not to break. By making the preservation policy explicit, the provider is exposed to the pressure by potential future users, who may eventually demand that the policy is enforced.

Technology can support such policies, for example by providing a mechanism for depositing commitment statements whenever a PID is issued. As Kunze further points out, this has been realized as part of the ARK system; still, persistence in the end depends on the participating parties obliging with the social policies. Even if they are made explicit by technological means, they cannot be enforced by them. At a larger scope, this resonates well with a common misconception in computer science and technology debates — the often naive assumption that technology (and its progress) will be a sufficient solution for all problems, blandly ignoring the human factors.

2.5 CONCLUSIONS

From the literature it is evident that the idea of persistent identification emerged out of practical needs. As a result, the definitions and more conceptual literature dealing with PIDs still cling very

much to the behavioral aspects and aim to define the concept by describing what you can do with a proper implementation of it: resolve identifiers over time and location changes. One could argue that the definition of identifier persistence stated as denoting (giving a name for) an entity indefinitely is just the same as the idea of identification per se: Every identifier is persistent. This however appears to be untrue for real-world objects. A simple example may be the name for a mountain or an important archaeological artifact. There is no guarantee that these names stay forever, since society evolves and so there is always the possibility for a change in name.

The contemporary notion of a persistent identifier is therefore highly bound to the technical background its applications stem from. Making an identifier persistent denotes a clear practical purpose (use case): Ensuring that the same entity is identified for a longer time span (indefinitely, in theory). Because the distinctions are purely based on such a behavioral view, it might be impossible to define a distinct conceptual class of persistent identifiers as a subclass of identifiers. The difference may be purely in the pragmatic behavior and as such, another definition cannot be found. If one was to subtract all the practical purposes, requirements and intentions, there may be no difference between a persistent identifier and any kind of identifier.

Therefore, PIDs are a social construct. This is the most important characteristic distinguishing a PID from primary keys in a generic relational database or the entries in a large distributed hash table. Technically, there is not a lot of difference, other than the specific use of information values that redirect to final objects. The Handle System, for instance, is an enabling technology that may make it easier to uphold the social construct, while technically, it is a set of interconnected hash tables. The policies constructed around the look-up and redirection facilities are thus essential. They define the responsibilities, regulate the issues of persistency, establish contingency plans so the application value of PIDs can be ensured over long time spans.

The brief history of persistent identification already shows a detectable shift in PID usage from simple redirection to storing additional context information closely associated with them. Technically, providing context information can be achieved through conventional databases, yet the social frameworks must evolve as well so they apply to maintenance of both identifiers and associated information. The literature also indicates a long-standing academic conflict between the schools of web-based URIs and the PID concepts favored by memory institutions such as libraries and archives. Attempts to unify these conflicting schools of thoughts have been made, yet as of today, the discussion appears far from being consolidated.

In general, the view that technology alone cannot overcome the fundamental issue of breaking links but that policies are required is important for the effectiveness of practical solutions. However, there are technological choices which make preservation easier or harder to achieve. A solution which makes the possible breakage of links a standard scenario should offer technical functions that support adequate policies.

3

USE CASES

As stated in the motivation on page 4, a framework for persistent identification must take into account various user requirements, including concerns of data curation and long-term preservation, the needs of e-science infrastructure maintainers and the original data providers. This chapter therefore introduces a set of practical usage scenarios that cover different user aspects but also indicate current issues in the practice of PID usage and scientific data management. Some of the scenarios are extensions of contemporary practice that hint at future developments. Since PID assignment and curation are accompanied by additional costs, the usage scenarios also indicate areas where added value can be provided by further exploitation of the identifiers assigned.

3.1 THE EARTH SYSTEM GRID FEDERATION

After many years of development and maturing, the Earth System Grid Federation (ESGF) [100, 20, 99] is today an open source project organized collaboratively by the global Earth system modeling community. The continued dependency of the CMIP5 and IPCC processes on it and the future perspective for CMIP6 have guided its development and shape the requirements and, last but not least, the goals associated with funding. ESGF is based on a distributed and federated architecture currently consisting of more than twenty nodes at individual modeling centers all around the globe. Figures 3 and 4 show exemplary screenshots of the ESGF end-user website and illustrate a data selection workflow..

ESGF is commonly considered to be situated at a dissemination stage of the data life cycle, situated after initial generation of data and partially reaching into archival and re-use stages. Some data served through ESGF is provided by long-term archives, however this is not necessarily the case for all data and most notably was not true during the evaluation phase of CMIP5 data. The use cases further below describe the ESGF process in more detail.

Due to the modular system design, ESGF nodes may differ in the set of features offered. A typical node will however contain at least a data services module to publish local datasets. Publication here means to make it visible within the federation, which is not the same as formal data publication with a DOI, which usually happens at a much later stage as described further below. Aside from the CMIP5 data, a lot of other modeling and observational data

Figure 3: The Earth System Grid Federation portal websites provide faceted search facilities. In this example from the ESGF node at `http://esgf-data.dkrz.de` (last checked Feb. 27, 2015), the currently selected facets are shown in the top-right corner: CMIP5 data from an Earth system model used by the Max Planck Institute for Meteorology, CMIP5 experiment abrupt4xCO2, monthly atmospheric data. The four entries displayed are consecutive versions of the same data with the final version at the top.

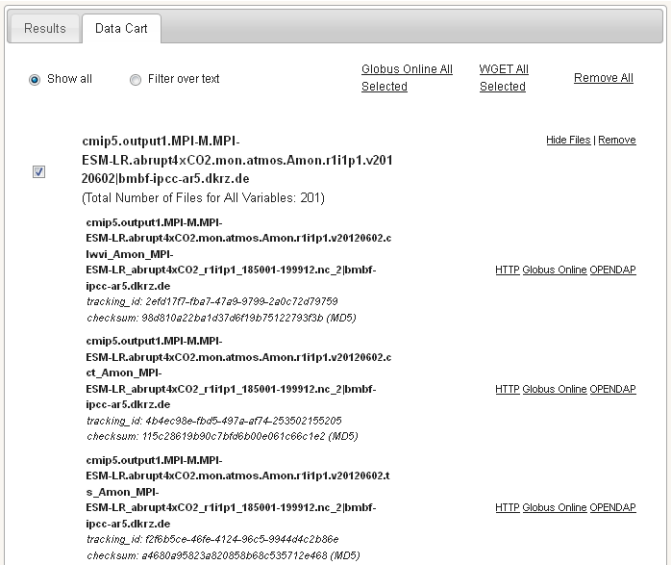

Figure 4: In this example, the first dataset from the result list in figure 3 has been selected. The dataset consists of 201 files (3 shown), a total data volume of about 50 GB. Each file bears a tracking ID and a checksum (see text). Additional metadata (such as the size) can be retrieved for datasets and files (not shown). The dataset is part of an experiment data publication bearing a dedicated DataCite DOI [33], which is one quite coarse aggregation level above the dataset level.

projects have recently decided to employ ESGF as a distribution infrastructure. Examples include modeling results of the WCRP-endorsed Coordinated Regional Climate Downscaling Experiment (CORDEX) and observational data from satellites (e.g. the "obs4MIPS" datasets). Typically, netCDF files will be published, and the metadata contained in their headers will be fed into the local metadata catalog. At every node, data can be searched, and the search will be federated across all nodes, so that in the end, remote datasets are visible in addition to the ones published at the local node. Access to them and download options will however only be given at the hosting node. Some datasets (e.g. CMIP5 core data) are replicated to more than one node to gain more efficient access by local users. A central access point to the ESGF data space are the web pages at the individual node, which offer complex searching and browsing facilities over the published datasets.

The following sections contain descriptions of four exemplary use cases from the ESGF application domain that may also be applied to other scenarios. The core use cases of ESGF, such as data publishing, searching and browsing, can already be handled by existing ESGF services and are not described in further detail. Similar use cases have also been described for example in [98, 97, 102].

3.2 REFERENCING PRELIMINARY DATA

The full process for eventually publishing a final dataset within ESGF is quite complex and covers far more than the technical publication aspects. A significant part of the process is subject to extensive quality control (QC) with several stages [85]. As part of these QC checks, both technical and conceptual aspects of the data are assessed. Technical aspects include for instance whether variable values are within valid bounds and whether the metadata entries are complete. At the final stage, the original authors of data must approve of the publication, and only after this has been done, the dataset is considered stable and a proper DataCite DOI is assigned. Due to the complexity of typical Earth system modeling outputs, the full process from first preliminary publication of a dataset on ESGF until DOI assignment can take considerable time, ranging from several months to more than a year. Errors in initially submitted data are quite common and in fact a central purpose of preliminary publication is to enable sharing and peer-review of data in order to correct them. In the worst case, whole simulations may eventually need to be re-run on the respective computational machinery, stalling the final publication process for considerable time.

However, scientific activities move at their own pace, and particularly the submission deadlines for the IPCC reports are tightly set. Scientists wish to refer to the preliminary datasets in an unambiguous way when discussing results with their peers or when preparing and submitting articles in time so they can be considered for the IPCC working group reports. In terms of Treloar and Harboe-Ree [90], such cases happen at the boundary between the private and the shared research domain. Yet, as long as a dataset is not ultimately stable and subjected to digital preservation, a full DOI cannot be assigned, as doing so would violate a fundamental policy of the DOI process, namely to keep objects available on the long-term. At creation of such an object it is unknown whether it will undergo the full life cycle or be disposed before reaching the archival stage. Short of making drastic changes to DOI policies, an alternative to the current usage of DOIs is thus required. A possibility for this is an identifier that is subject to different policies than a DOI, however it is unclear how and in what respect such an identifier can be considered persistent if the dataset may be withdrawn and which other consequences may result.

3.3 ACCESS TO SPECIFIC VERSIONS OF A DATASET

A further variation of the problem described above concerns the versioning practices of ESGF data. As shown in figure 3 in the context of CMIP5, there can be multiple versions of a dataset published over the course of several month. Imagine that indeed publication of a

dataset has been stalled after the initial stage because some errors were detected or some files were still missing. Now the dataset has been fixed by for example reprocessing still available raw data or re-running the model. In the meantime, a user may have already used the data, such as for further processing and analysis. Before publishing an article, such a user should obviously check whether the data used contain any errors discovered after initial retrieval. The user may now submit the data files to a specific service or extract some information from them in order to look up the associated information in ESGF and ultimately discover that there actually is a new version of the dataset. The user may choose to retrieve the latest version, and in any case the user will want to know when and why the original data have been replaced.

This scenario can be further generalized outside the scope of ESGF. Data that are subject to frequent or infrequent changes should be uniquely identified across these changes, so that there is always a method available to retrieve the latest version of the data through a dedicated persistent identifier. This is useful for example for continuous processing of real-time sensor data. At the same time, however, it is also important to keep references to older versions and make them discoverable so that third parties can investigate possible changes.

3.4 REFERENCING CUSTOM DATA SLICES

Earth system models generate complex data products that cannot be easily categorized along a single hierarchy. Although it may seem to be a natural way to arrange all CMIP5 data along the organizational axis top-down from experiments over simulations to the time series of individual variables (see figure 4), this does not reflect the manifold axes along which different users understand CMIP5 data products. For searching data at ESGF nodes, the issue is mitigated through a faceted search interface at the node websites (see figure 3). However, the purpose of model intercomparison is to mix and match data from various models, simulation runs and modeling centers, and also possibly include observational data. Referencing such a colorful mixture of data both completely and precisely is quite difficult given the current DOI assignment and usage policies. There is a certain practical limit to the number of DOIs citable in a reference list, and often, these DOIs are assigned at a quite coarse level of granularity and for a thematic grouping not adequate to precisely define the base data of the particular work conducted.

Ideally, a user will therefore fill a “data shopping basket”: Before submitting an article on an intercomparison analysis of model results, the user presents all data used as a big bundle to a specific service. The service verifies that all data are still available and of the latest

version. It then provides the user with a single unique identifier for the bundled data. At a later point, looking up this identifier allows a third party to assess exactly which datasets have been used. It should also be possible to request more information on them, such as whether data are in a stable form and if not, whether (and where) there are new versions of them. Ultimately, it may even be possible to recompute any results in the article if the shopping basket contains all input data and references a proper executable workflow. Such data reproducibility aspects are discussed further below in the context of provenance.

As a further expansion of the idea, the data basket from a model analysis effort may even contain products from different e-infrastructures, such as model output data from ESGF and observational data from the Global Earth Observation System of Systems (GEOSS) portal. Even if all data bear PIDs, they may be issued through different PID systems depending on what the particular e-infrastructures use. A solution will therefore ultimately have to bridge PID systems as well as e-infrastructures to enable users to reference the full context of the scientific work conducted.

As a further generalization, the use case of referencing a custom data basket is not bound to ESGF or the geosciences. Ultimately, the service to receive a data basket and assign a PID to it may be offered by scientific publishers that ultimately also govern the acceptance of PIDs in articles if not in reference lists. Of course, the publishers will try to avoid dealing with individual data objects and information about them (e.g. displayed on specific landing pages) themselves. This can however be easily delegated to the original data holders as they also provide the individual PIDs aggregated in the basket and will thus most likely offer rich information on their own.

3.5 PROVENANCE TRACING

Another use case that by its very nature spans several infrastructures such as EUDAT, ESGF and other community-specific solutions concerns cross-system provenance tracing. Provenance information forms an important part of scientific metadata as it facilitates quality and validity assessments of data and allows third parties to understand the processes which lead to the creation of a particular data object [65, 67, 82]. Research on provenance-related topics has produced significant output on topics such as provenance encoding formats [68, 66] and assembling provenance information from scientific workflow engines [81]. A particularly useful view of provenance as described by Moreau [65] is to model it as a directed acyclic graph of data objects that are derived from each other.

When a data object is re-used and processed, it may be published at a location different from the original repository. If this happens

multiple consecutive times, the provenance trace of the object runs into a high risk of becoming interrupted if one of the intermediate repositories becomes unavailable or if objects are deleted because they were not covered by a long-term archival policy. Even if there are provenance documents e.g. produced by a workflow engine, these will become inaccessible if the cross-repository trace is broken. Therefore, a solution is required that allows users to trace back the acyclic provenance graph of data objects across repositories and temporary states that may already be gone at the time of investigation. As also mentioned in [98], it is preferable to ensure some form of *graceful degradation*, meaning that upon partial information loss, as much as possible of the remaining provenance trace and context information remains meaningful.

Duerr et al. [26] also state that indeed there may be value in keeping some context information intact even if the data objects are already gone, justifying the additional preservation costs. Ultimately, rich provenance information may enable the recomputation of data products from the original inputs, particularly if data have been produced by means of a scientific workflow system that fully captures all production steps in machine-interpretable form (exemplary solutions are presented for example by Van Gorp and Mazanek [93] or Koop et al. [50]). This goes beyond the scope of this simple use case, where the user only wants to discover the original sources that were used in creating a final data product, even if they are located at separate repositories. Nonetheless, discovering the initial inputs and intermediate data products can help users to acquire workflow descriptions that eventually enable recomputation. With additional information available on intermediate products, such as checksums, the execution results may also be verified.

3.6 DISCUSSION

For resolvable PIDs, the most important use is resolution. Kahn and Wilensky [45] describe this as a process where a user submits a PID to a resolver and the resolver will return the network names or addresses of repositories in which the digital object is stored. Ensuring reliable resolution of identifiers over long time spans is a fundamental requirement across all use cases presented here, even though they illustrate usage beyond simple resolution.

Not only in ESGF, but also in application scenarios from other European e-science infrastructures, such as the Common Languages Resources and Technology Infrastructure (CLARIN) or the aforementioned Digital Research Infrastructure for the Arts and Humanities (DARIAH), there are recurring questions of how to deal with PIDs when they occur in massive numbers, are used across infrastructures and last but not least identify different kinds of entities: there are

already solutions for identifiers for scientific articles and data objects and more recently also for people, organizations and geological samples. Machine agents acting in infrastructures designed to deal with massive numbers of data and other objects will be unable to properly perform automated tasks if they do not know the essential properties of the resource behind an identifier, most importantly what major category it falls into. An identifier is supposed to be an opaque string (cf. [72]), thus the resource will have to be analyzed to provide the necessary context for the agent to act. But such a solution does not scale well if large numbers of objects must be assessed in a limited time frame and it is also unsuited for objects stored at remote locations as is typical in the context of contemporary distributed systems. In digital preservation, there is also the notion of dark archives where access is impossible or extremely difficult and expensive. The context of the resource must therefore be available in a rapid fashion and it should be sufficient to define the fundamental type or class of a resource.

4

CONCEPTUAL FRAMEWORK

This section first introduces a formal model that defines fundamental notions such as an identifier, PID systems and PID records. The formal model provides the basis for a conceptual framework for identifiers and associated information. The framework consists of three abstraction layers, where each higher layer relies on functionality provided by lower layers (see figure 1). If an upper layer is lost, the information and operations provided by a lower layer must continue to be useful and coherent. Applications may choose to work on top of any of these layers, depending on the complexity of tasks that need to be accomplished.

The bottommost layer is called the *persistency layer* because it ensures that the most fundamental aspects of persistent identifiers and the relations expressed in the formal model are obeyed. It also offers a set of criteria useful to classify different PID systems. Its top interface is provided through the concept of a Persistent Entity that is essential to define how identifiers can remain persistent if their objects are not preserved and how essential information can remain available beyond the lifetime of an object. The *typing layer* describes PID records in more detail with the goal to enable the classification of objects by typing the information within their PID records. The *collection layer* finally steps beyond a view of singular identifiers and objects by providing persistent mechanisms for aggregating objects and enabling efficient operations on such aggregations.

Since practical applications may not require all facilities offered by higher layers, two broader models are referred to throughout the framework. In the *full model*, all layers are present. The simpler *preservation-centric model* may be easier to understand and applied by users interested purely in digital preservation aspects: It focuses on the Persistent Entity and does not include the aspects of typing and collections. It is up to the reader to decide which model fits a particular purpose best; in any case, both share the same foundations.

4.1 FORMAL MODEL

The formal model aims to describe the relationship between identifiers, objects and PID record contents. The relations defined in the model also cover changes to object location and PID record contents over time. A more detailed model may describe such changes in further detail, for example through sequences of discrete states, yet the level of detail pertaining to such a view is not required to

address the initial research questions and lay the foundation for the framework layers described through the course of this chapter. Also note that the question of how exactly object identity is defined is considered to remain open with two possible answers: Objects may be bit-identical, sympathetic to a view of bit-stream preservation, or considered to be equal from a domain sciences viewpoint, which involves curative actions such as file format migrations. This is further discussed in the context of criterion 6 further below.

4.1.1 *Describing identification*

To describe identification as the basic element of the formal model, let us begin with a very simple world view where there is a finite set [36] of data objects, D, and a finite set of identifiers, P, with $|P| \geqslant |D|$. The goal of identification then is to construct a relation between these two sets. Let us name this relation $D\phi P$, circumscribed as "an object is identified by an identifier". To be practically useful, ϕ will however have to fulfill some properties.

The most essential property originates from the problem that an identifier is not very useful if it identifies more than one object. Specifically this is what it means to be a *unique identifier*: there should not be two pairwise distinct objects being identified by the same identifier. In consequence, ϕ must be injective.

Definition 1 (Unique identification relation) *Let* D *be a finite set of data objects (bit-streams). Let* P *be a finite set of identifiers. Then an injective relation* $D\phi P$ *is called a* unique identification relation.

The injectivity of ϕ is described as follows:

$$\forall c, d \in D,\ \forall p \in P\colon\ c\phi p \wedge d\phi p \Rightarrow c = d \tag{1}$$

The definition of the unique identification relation allows us to have a look at another, stronger interpretation of uniqueness. In practice, it may ultimately be desirable to assign not more than one identifier to any particular object (as is, for example, described in [21, 6]). This can be defined as follows: A unique identification relation is *strictly unique* if and only if it is functional (univalent). Note however that such a scenario can hardly be enforced in practice and may also have undesirable consequences. In the following, we will therefore only assume simple (non-strict) uniqueness of ϕ unless explicitly stated otherwise. The Entity Name System presented by Bazzanella et al. [6] aims to enforce strict uniqueness, however it exhibits some fundamental flaws which are discussed later in section 7.2.

There is also an equivalent view. Consider the following logical statement: If two identifiers are identical, then their referenced

objects are identical. This is equivalent to the non-strict uniqueness interpretation. The strict uniqueness is expressed as an equivalency: If and only if two identifiers are identical, then and only then their referenced objects are identical. This allows us to infer from the uniqueness of objects to the uniqueness of their identifiers. In practice, such a strict interpretation often cannot be assumed.

For practical purposes, we also often do not want to look at ϕ and its properties, but rather at its inverse relation, $\phi^{-1}\colon p \longmapsto d$, paraphrased as "an identifier identifies an object". This relation is essential for practical retrieval of an object if we only have an identifier. This directly brings us into the scope of a PID system:

Definition 2 (PID system and identifier resolution relation) *Let* D *be a finite set of data objects. Let* P *be a finite set of identifiers. Let* $D\phi P$ *be an injective unique identification relation. Then* $S = (P, D, \phi^{-1})$ *is called a* PID system *with* P *as the set of identifiers of* D. $P\phi^{-1}D$ *is the inverse of* ϕ *and is called the* identifier resolution relation.

Since ϕ^{-1} is the inverse relation of ϕ, it follows from (1) that ϕ^{-1} is functional, also called a (partially defined) function [77]. Using (1) with the inverse of ϕ gives us the common definition of a functional relation:

$$\forall p \in P,\ \forall c, d \in D\colon\ p\phi^{-1}c \wedge p\phi^{-1}d \Rightarrow c = d \tag{2}$$

The value of ϕ^{-1} can be undefined in cases where the data object d is already gone but the identifier is preserved, thus ϕ^{-1} may be only partially defined. Vice versa, if ϕ is strictly unique (i.e., functional), ϕ^{-1} is also injective. Finally, the goal of *persistent* identification then is to control changes to ϕ^{-1}, essentially minimizing them. This will later be expressed in the persistency layer as the first criterion.

4.1.2 *Describing PID records*

In the simplest understanding, a PID record is a mapping — a function — from keys to values. To keep things simple, we use natural numbers as keys and leave the exact definition of values open without loss of generality.

Definition 3 (PID record function) *Let* V *be a set of values. Then a partial function* $\omega\colon k \longmapsto v$, $k \in \mathbb{N}$, $v \in V$, *is called a* PID record function.

We will examine the definition and properties of ω in more detail and see that in general we cannot assume it to be surjective or injective.

Let us assume that a particular key should only point to at most one value, because that is the foremost use of PID records for access

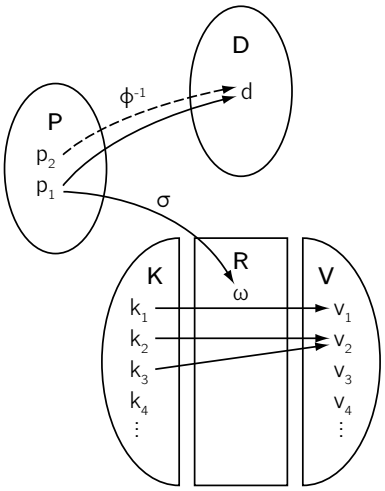

Figure 5: Sets and relations for an extended PID system. P is the set of identifiers, D the set of data objects, ϕ^{-1} is the identifier resolution operation. In case of non-strict identifier uniqueness, ϕ^{-1} need not be injective: both p_1 and p_2 can be related to the same d. PID records are described as members of the function set R, an example for which is ω. The PID record resolution relation σ associates a PID with a PID record. ω may be partially defined: There is no image for k_4. ω need not be surjective: v_3 and v_4 have no equivalent in k. ω need not be injective: k_2 and k_3 map to the same value v_2.

and storage of information[1]. Consequently, ω should be a function. However, in practice, the size of PID records, i.e. the number of keys in a particular record, is quite variable, and not every PID record will contain a value for every key; thus, ω is usually only partially defined. Also, different keys may map to the same value, perhaps because the dates for its creation and last access are identical; thus, ω is not necessarily injective. The image of ω may also be a very small subset of V because of the large number of values possible (e.g. character strings of arbitrary length), thus ω is not necessarily surjective.

Essentially, the notion of a PID record is more precisely expressed as a concrete instance of a PID record function.

In much the same way as, given an identifier, the resolution relation provides us with an object, we can also describe a similar relation for PID records. By extending the notion of a PID System, it becomes possible to handle identified objects and PID records based on a common structure.

Definition 4 (Record resolution relation and extended PID system) *Let* $\mathbb{S} = (P, D, \phi^{-1})$ *be a PID system. Let* V *be a set of values and* R *be a set of PID record functions. Then a functional relation* $\sigma\colon p \longmapsto \omega$, $p \in P$, $\omega \in$

1 Other scenarios are possible where e.g. a record contains multiple text fields, all keyed to be a "description". ω will then not be functional.

R, *is called a* PID record resolution relation. $S^+ = (P, D, V, R, \phi^{-1}, \sigma)$ *is an* extended PID system.

The relation σ is functional (within the scope of a distinct PID system) to ensure that there are not multiple records stored for a single identifier. Note that the identifier p is in general not part of the PID record. Thus, the relation σ is not necessarily injective: two different identifiers can be associated with the same PID record (i.e., the same values as expressed by the mapping function ω). This may particularly be the case if there is a "default" or "empty" record. Note that σ is not required to be a total function at this point, covering scenarios where a PID record is unavailable even while the identified object still exists.

All sets, relations and functions described so far are also summarized in figure 5.

4.1.3 *Further aspects*

Duerr et al. [26] define the purpose of unique identification as to identify an object unambiguously, no matter which copy a user has. This leads to the question of how to define identity and determine differences. If there are multiple copies of an object, then the defining difference between the copies is their location. For this reason, Duerr et al. require that an identifier (as opposed to a locator) is independent of location.

On the other hand, there are use cases which require identification also of different copies of the same object. A good example for this is replication: even though replicas are just copies of the same object, bit-identical with it, their location is different, and typical use cases requires that each copy is uniquely identified for purposes of its management locally at the replication repository. What is needed at this point is a mechanism that enables identification of the object as a whole, being able to determine its identifier from any copy and identification of each individual copy including a mechanism to locate it (for example for verification). Such a mechanism can be enabled through adequate use of collections as practically discussed in chapter 6.

The relationship between the PID record and metadata is quite complex, and the formal model alone cannot address these adequately. Generally, the information stored in the PID record is part of an object's overall metadata. However, not all metadata is suitable to be included in a PID record. Section 4.2.3 further discusses such metadata aspects.

There are also viewpoints that state that the distinction between data and metadata is artificial because someone's metadata can become data for purposes of statistical analysis, recombination and so on. This is however very much depending on the specific discipline

considered. For the Earth system modeling community, there is a very differentiated and clear distinction between data and metadata; for purposes of clarity, the remainder of this thesis uses the expression "domain metadata" where applicable.

4.2 PERSISTENCY LAYER

The use cases demonstrate that there are various applications for context information closely associated with persistent identifiers. Two major themes characterize the information possibly stored in PID records:

1. PID records are built on the idea of *rapid resolution*: Associated context information is available from a high performance system and without having to retrieve the identified object that may be large and reside in a system with less efficient access. The target object may also be a metadata object, which contains a lot more sophisticated and semantically rich information than the PID records at additional costs that should be avoided in favor of rapid look-up.

2. There is value in keeping some information beyond the lifetime of the identified object, so that it is persistently available given an identifier and an adequate resolution system (cf. [26]).

One important question is whether these two fundamental use cases cover the same or at least similar kinds of information. Another question is how to organize the information in a way so that it is both suitable for rapid look-up and interpretation and maintainable on a longer time scale to address the need for persistency. The persistency layer is thus situated at the challenge of providing some first answers, while a full understanding will emerge through the full layer model. The persistency layer is characterized by following a view focused on single PIDs, PID records and objects, and ensuring the adequate access to information according to the two goals of rapid resolution and persistent access to PID records and potentially resources. A notable fundamental requirement and base assumption important for later implementation is that PID record information is accessed through keys, and that access can be served in constant asymptotic time. This is in line with the earlier definition of the PID record function.

A central idea of the persistency layer is to define criteria that can be used to classify PID systems. The criteria defined further below are based on strict requirements initially described in [98], however these particularly lack the formal foundations presented earlier, although they remain valid for their particular use case. A view that makes the criteria explicit as strict requirements, obligatory for any PID

system, would not adequately reflect that different approaches can co-exist with varying sets of requirements as shaped by their particular use cases. It is therefore useful to reformulate the requirements as a choice of criteria. A PID system may then conform to any number of these fundamental criteria, effectively using such a subset as requirements without demanding conformance to all possible criteria. Certain combinations may still turn out to be more useful than others, which is reflected in three archetypical scenarios defined in section 4.2.2.

The criteria are not of a purely technical nature; several of them can only be met by enforcing certain policies, which was also an insight described by Paskin [73]. A PID system is therefore a conceptual entity consisting of a known number of concrete resolution services, a defined schema for PID names and a set of policies that are enforced by the resolution service provider. This is not in conflict with definition 4. Naturally, the resolution services should only resolve identifiers considered persistent, or, in other terms, the user expects any identifier successfully resolved through the system to be a persistent one. Most notably, this excludes the Domain Name System as a top-level resolver, because it resolves URLs that are not considered to be persistent per se (see section 2.4.2). The naming schema must define the syntax for identifiers, allowing human users or machine agents both to check any given identifier for conformance and to generate new names. It must also be specific enough so that a user will know how to dereference the identifier. This is mostly a question of adoption: today, DOI names are encountered by a large enough number of scientists that the question of how to dereference them is easily answered (e.g. through a short web search).

There are two fundamental types of resolution operations enabled by a PID system as also described in [98]. Given an identifier, a *resource resolution operation* will return the target object. In practice, this operation exceeds the scope of the PID system in the strictest sense, since it also relies on the actual storage mechanism for the object. A *record resolution operation* will return the contents of the PID record associated with the identifier. These two operations can be affected by the fundamental criteria.

4.2.1 *Fundamental criteria*

There are different approaches for designing and using PID systems and different expectations regarding their reliability. The criteria can be used to classify existing approaches (also see section 5.1) and as requirements they shape the technical solutions and management policies of PID systems to maintain a desired level of quality. Note that all criteria must be interpreted in terms of a single particular PID system, because the namespace of identifiers P is always bound

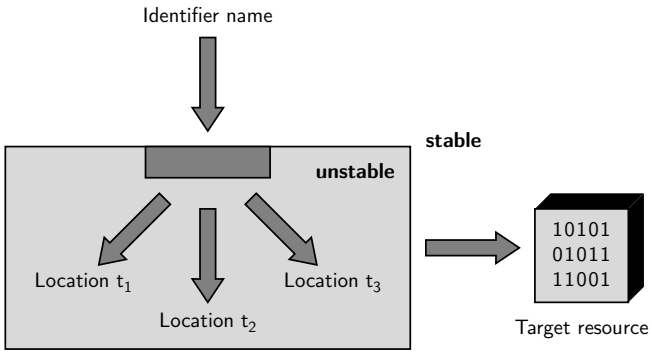

Figure 6: PIDs establish a layer of redirection: From an external point of view, a change in resource location does not affect the resolution, it is *stable*. The internal logistics are *unstable*, separated from the external view through an abstraction or interface.

to a particular system to ensure uniqueness. Also, all criteria only apply to resolvable identifiers: There must be at least one resolution service.

Criterion 1 *The link to the resource that an identifier identifies can be changed: If the resource is relocated, it is possible to reflect this for the identifier in the resolution services without changing the identifier name. The resource resolution operation continues to be performed successfully.*

This criterion is formulated more precisely as the corresponding requirement in [98, p. 17] which did not further specify the relationship to the identifier name and resolution services. The criterion captures the most essential characteristic of a persistent identifier; a system not conforming to it is in fact not a PID system. Vice versa, a system conforming only to this criterion is a PID system, making the criterion a necessary and sufficient condition.

To further understand the criterion, consider a non-persistent identifier. This identifier may be resolvable at some point in time, but if the identified object relocates, it will be impossible to retrieve the object, even with the help of a resolver service if that service knows only the former address of the object. The most essential characteristic of a persistent identifier is thus that it continues to be resolved through changes in object location or ownership (also see section 2.4.2). This can also be expressed as the metaphor that PIDs establish a layer of redirection (see figure 6). In terms of the formal model, the criterion aims to ensure the integrity of ϕ^{-1} over time. Note that ϕ^{-1} associates identifiers with data objects, independent from their location; as explained earlier, a more detailed model would formally encode the changes of location, which is inherent in the concrete expression of ϕ^{-1}. The first criterion simply aims to maintain the association from p to the same d.

Criterion 2 *A registered identifier is globally unique: The result returned by a successful resolution operation of the same registered identifier is identical for any two non-identical resolution services.*

This criterion is equivalent to the corresponding requirement given in [98, p. 17] as it captures a fundamental user expectation. If a user encounters an identifier and submits it to a resolver, it should not matter where or how this identifier was encountered and consequently which resolver it was submitted to — the user expectation is that it should resolve to the same result independently of those factors. In terms of the formal model, fulfilling this criterion requires that the manifestation of ϕ^{-1} does not vary within a particular PID system.

If the PID system offers the ability to define multiple namespaces (and the current or expected usage is of course to employ more than a singular namespace), some issues must be analyzed to be sure the criterion is fulfilled. Namespaces are defined by the PID system's naming schema. If there is more than one namespace, then a precise indication which namespace a particular identifier belongs to is required, and thus the identifier naming schema must include this as a mandatory part. Otherwise there may be identical identifier names that belong to different namespaces but this will not be visible solely from their names.

This also demands that either the location of a resolution service must not be part of the identifier name or that every resolution service can resolve any identifier that is bound to a foreign resolver, e.g. by forwarding requests.

Note that the criterion specifically addresses successful resolution operations. If a resolution operation is unsuccessful, but the identifier is registered, then there will be at least one other resolver that will resolve it successfully, albeit the user may not know its address. The essential motivation of defining this criterion is therefore not to evaluate alternative resolution paths, but to judge whether there is coherence among the results of resolution operations by decoupling the questions of identification and resolution from the particular user context.

Criterion 3 *All record resolution operations continue to succeed, even if a resource becomes unavailable.*

Aside from the notion of a PID record instead of generic metadata, this criterion is equivalent to the requirement from [98, p. 18]. This is the criterion that characterizes the specific behavior desired for records being directly associated with identifiers. As explained earlier, only a specific usage model of identifiers will require records to be kept past an object's life time. If records are considered expendable once a resource is gone, fulfilling this criterion is actually undesirable.

A system striving to fulfill this criterion as an essential requirement can in consequence ensure that inter-identifier links remain resolvable

(i.e. to the PID records of their respective identifiers) even if both the source and the target resource become unavailable. Since practically no PID system will be able to track the global usage of its identifiers, meeting the criterion may be considered an integral part of the service concept defined in any system's policies to uphold the trust in all identifiers served through it. Even if a resource is deleted, causing future resource resolution operations to fail, there will still be a record available by querying the PID system. This also implies that no identifier can be purposefully deleted (unregistered). In practice, this can of course happen accidentally or during test phases.

In the formal model, this criterion describes the record resolution relation σ and how it remains unaffected by possible changes to ϕ^{-1} over time.

Criterion 4 *All registered identifiers are globally discoverable: There is at least one globally accessible resolution service that directly or indirectly resolves all registered identifiers.*

This criterion is equivalent to the corresponding requirement from [98, p. 17]. To understand the motivation for this criterion, imagine how a machine agent will perform tasks. The agent aims to resolve a particular identifier but lacks any further information, such as its original issuer, and cannot deduct this knowledge on its own. Let us assume that the criterion does not hold, i.e. there is no central resolution service, and that the agent has a list of resolvers. Then it cannot be assumed that this list is complete: Since resolvers can change location or be decommissioned, it may be practically impossible to maintain a complete list for every agent dealing with identifiers. Thus, if a resolver responds with a failure upon resolution, the machine agent can only assume that this particular resolver does not know the identifier. Even if the agent queries all resolvers from its list, it cannot be sure to have a definite answer because its list will be incomplete.

A single omniscient resolver whose location is known to all agents will solve the issue. There are also alternatives, indicated by the idea of indirect resolution, such as federating queries between resolvers that know each other or building a meta-resolver that supplies a single omniscient instance, possibly based on querying individual resolvers. However, the entity providing the meta-resolver will still have to make sure it has complete knowledge about resolvers. How to practically achieve this, e.g. through policies or federated algorithms, shall not be of concern here. The exemplary PID systems discussed in section 5.1 employ individual strategies to deal with this issue.

In terms of the formal model, this criterion demands that there is a resolver service which can provide all images of ϕ^{-1}.

The criterion also demands that policies are put in place that ensure that such a resolver continues to exist beyond the lifetime

of its maintainer, for example by transferring its responsibilities to a different resolver.

Criterion 4 can be extended:

Criterion 5 *Any globally accessible resolution service that successfully resolves one identifier is able to resolve any registered identifier.*

This criterion is equivalent to the corresponding requirement from [98, p. 17], however lacking the explanations as given by the formal model. Conformance to it can be described colloquially as requiring that all resolvers (of the same PID system) behave in the same way, each of them providing all images of ϕ^{-1}. A system fulfilling this criterion may be designed in one of several possible ways.

There may simply be just one resolver (or a handful of identical mirrors) that redirects requests to a known set of more specialized nodes. The effort then lies in maintaining the nodes or the association with them in case they are not unified under an umbrella organization. This may be done for example by maintaining a list of all known nodes at the central resolver, and putting policies in place that control the introduction and decommissioning of nodes. As also stated for criterion 4, this may be practically impossible to uphold.

Alternatively, there may be many resolvers, which will appear to be redundant from a functional point of view. However, such redundancy may be desirable for load-balancing and reliability purposes. It is also beneficial for users unfamiliar with the PID system. They may be unaware of a change in resolver ownership or location, or they may be used to a particular resolver, perhaps set up for their individual project, and will be confused if they attempt to use an identifier external to their project with this resolver, expecting it to be resolved but getting a failure. Making all resolvers equivalent as to the set of identifiers they can resolve will eliminate such problems.

Note that it is not possible to fulfill this criterion without fulfilling criterion 4 as well, making criterion 4 a necessary condition.

Criterion 6 *A successful resource resolution operation returns the same response given the same identifier indefinitely.*

In contrast to the requirement given in [98, p. 18], there is no notion of a separate record resolution operation purely aimed at immutable metadata here. To fulfill this criterion, a resource must not be modified once an identifier has been registered for it, which is a requirement essential to digital preservation. As also described in [98], if a registered identifier is ever deleted (unregistered), it may never be reassigned to a different resource. Again, it is important to remember that it is not desirable to meet all criteria for all use cases, and this is a particularly good example requiring careful consideration. If interpreted as a strictly enforced requirement (e.g. through adequate policies), it may have unwanted consequences. For

example, conformance implies that if a resource is versioned, i.e., if a new or revised version of the resource is published, the old identifier cannot be reused. Conformance to the criterion does not demand however that resources are never deleted — it only covers successful resolution operations. In terms of the formal model, if a resource is deleted, the particular d associated with p may become irretrievable, yet conceptually it remains identified. A more detailed formal model may be able to describe the particular point in time when such an event occurs.

As also pointed out in [98], this of course raises the question of how the identity of resources is defined and, consequently, the difference between resources. As mentioned earlier, the formal model remains agnostic with respect to constraints pertaining to specific domains and use cases. In the area of digital preservation, for example, distinctions are made between bit-stream preservation, where any change to stored data is disallowed, and approaches that aim to provide long-term readability, which may involve format conversions. From an infrastructural, technical perspective, a notion of bitwise identity may appear beneficial because it is easier to assess, but this is not always desired. Each scientific community may have its own interpretation and existing policies for defining resource identity, and for this reason, the exact answer of object identity must remain open at this point.

4.2.2 *PID system classes*

If the criteria are interpreted as requirements, it is possible to define general classes of PID systems based on which criteria are minimally fulfilled. The motivation for each of these system classes emerges from practical usage scenarios that emphasize differing aspects. The criteria determined for each type are the required minimum for the desired functionality; of course, a particular system may fulfill any number of additional requirements.

A PID system is **automatable** if it conforms to criteria 1-4. Its purpose is to be used by automated tasks or software agents with the goal to minimize the additional contextual knowledge required from them to perform their actions. An example is the Handle System if combined with policies that cover node decommissioning.

A PID system is **preserving** if it conforms to criteria 1-3 and 6. Its purpose is to ensure reliable access to both resources and records on a long-term scale. An example is the URN system if combined with policies that ensure long-term archival of resources, e.g. through libraries.

A PID system is **resolver-independent** if it conforms to criteria 1, 2, 4 and 5. Its purpose is to minimize the difficulty of resolution tasks by providing a unified access layer. An example is a distributed hash

table. Policies regarding preservation of resources or records can be minimalistic.

4.2.3 *Defining the Persistent Entity*

The foundational criteria frame the reliable context of PID systems. While there is a notion of a persistent PID record, the components introduced so far are only loosely coupled. This is adequate for many traditional applications of PIDs where there is no broader use of PID records beyond purely internal administrative aspects; yet, for more extensive use, a model is sought that establishes a more rigid frame in which to formulate persistency of metadata, objects and relations between them.

The top-level interface of the persistency layer is therefore described through two central entities: the Persistent Entity (PE) and the Digital Object. These also characterize the difference between the preservation-centric model and the full model. In the full model, the Persistent Entity and Digital Object act as the primary interface for the more sophisticated layers, while the persistency layer relies on a particular context made up from the criteria describe so far. Both Persistent Entity and Digital Object are defined based on the formal model. The operations of an abstract data type interpretation of the Persistent Entity are essential to the preservation-centric model because they provide the necessary facilities to differentiate between a primary and a secondary level of preservation. In [98], a similar concept for a Persistent Entity based on an understanding of it as an abstract data type has been presented, including the notion of two distinct preservation layers. This concept is extended here in accordance with the formal model and the layered framework.

Using the formal model, both a Persistent Entity and a Digital Object are precisely defined in the following way:

Definition 5 *Let* $\mathbb{S}^+ = (P, D, V, R, \phi^{-1}, \sigma)$ *be an* extended PID system. *Then every 2-tuple* (p, ω) *where* $\sigma(p) = \omega$ *is called a* Persistent Entity, *and every 3-tuple* (p, d, ω) *where* $\phi^{-1}(p) = d$ *and* $\sigma(p) = \omega$ *is called a* Digital Object *with* $p \in P, \omega \in R, d \in D$.

The value of $\phi^{-1}(p)$ has no effect on the Persistent Entity, and thus the defining mechanism — the resolution functionality — is covered by $\sigma(p)$; this relation must be preserved at the primary level, which is in accordance with criterion 3. We can also deduct that every Digital Object has at least one directly associated Persistent Entity. There can be several Persistent Entities associated with a Digital Object because there can be multiple identifiers for the data object (assuming non-strict uniqueness) and each of the identifiers may be associated with an individual PID record.

The definition has implications on the notion of equality of Digital Objects or Persistent Entities. If there is no assumption of strict uniqueness, i.e., if there can be several identifiers for the same data object, then these form individual Digital Objects. This also leads to situations where there are several identifiers with individual PID records associated with the same data object.

The initial motivation to formulate the Persistent Entity stems from a particular conceptual model associated with criterion 3. Some context information about a resource should be stored as persistently as possible and beyond the resource's lifetime; good examples are resource provenance and inter-identifier links to related context objects. If a system fulfills this criterion and the most valuable information is put in the PID record, a distinct *primary level of preservation* emerges [98]. The *secondary level of preservation* then consists of the target resources. The two fundamental resolution operations are directly associated with these levels. Through the separated levels of preservation, context information can be preserved even if the preservation of resources at the secondary level fails.

The primary level of preservation is particularly valuable when individual PIDs are not isolated entities: The more relations between PIDs are established at the primary level, the more useful the information becomes. This does not only cover the relations themselves, but also any other context information that consequently becomes discoverable as well. Section 7.1 discusses such aspects in more detail with respect to Linked Data.

The definition of the Persistent Entity provides the notion for an entity that exists beyond the lifetime of a resource and also extends the merely resolution-focused capabilities of the persistent identifier. The PID is simply an access tool: If the PID is understood as just the identifier name (and a tightly associated promise to keep it resolvable), then maintaining the identifier does not cover the context information. It certainly is an important factor, but not sufficient to cover the full scope of preserving the resource context. Speaking of "preserving the persistent identifier" is not an adequate formulation. Instead, the object that is preserved beyond the resource's lifetime and thus different from both the resource and the PID is the Persistent Entity. This is also not precisely the same as the PID record, i.e. the stored information. The PID record is essential to the Persistent Entity, but the PE must address more than the question of storage; most importantly, the PID record as a storage structure does not address the association with the identifier, the process of record resolution and the potential retrieval of the actual resource.

A more pragmatic interpretation of the Persistent Entity definition emphasizes its interpretation as an abstract data type (ADT) (cf. [98]). Because the Persistent Entity forms the top-level access interface for the preservation-centric model, it is beneficial to understand the

Persistent Entity as an actionable object, something that provides operations that machine agents can execute; this is not limited to the user's viewpoint of being able to use an identifier e.g. by calling it in a web browser as described by Kunze [52] and Paskin [72]. According to common textbook definitions [56], an ADT is purely defined by its operations; thus, in fact, the PID record is not a defining part of it, but rather a tool for implementing such an ADT. Other possible implementation pathways exist, e.g. through a separate meta-database and appropriate preservation policies. The fundamental criteria must however be respected; a Persistent Entity can only be served through a preserving PID system.

There is also a promise involved in the general motivation for the PE, namely to keep it resolvable through the PID. This is not the same as preserving the resource, because the PE is a distinct entity that can be preserved separately at potentially much lower costs. Considering the actual metadata provided via a PE, there is, in principle, no limitation to the type of information available. In the broadest understanding, it is entirely up to the generator of a PE what to provide. The use cases presented in chapter 3 give some indications. One distinction is that the Persistent Entity aggregates all information that is deemed to be important enough to preserve beyond the lifetime of the identified object or that must be available at a level of rapid access that is considerably faster than accessing the object or a separate metadata storage. In more detail, information possibly available from a PE falls into two broad categories:

DOMAIN METADATA provided by the original issuer of a PE. Individual scientific disciplines or communities typically design and use their own metadata standards to describe the scientific context of their data, occasionally in elaborate detail. On the other hand, some of this information may be targeted towards the interdisciplinary, useful for actors from outside the realm of the original issuer and thus perhaps encoded via cross-domain metadata standards or well-accepted ontologies. The more interdisciplinary such information becomes, the more useful it should be considered to be included within a PE. This information may not be required to be rapidly available, however.

OBJECT MANAGEMENT METADATA which enables some access and management tasks independent from knowledge about the domain meaning of the resource. A good metaphor for this is the 'envelope' metaphor of e-mail delivery through the Simple Mail Transfer Protocol (SMTP) or any other form of 'black box' metaphor (also cf. [95]): What has to be written on the outside of the container so any agent dealing with it knows how to perform a specific task without having to look inside, which

may be a very costly action? Typical interpretation tasks include to trace the provenance of the object (or another, possibly still existing object whose trace this object is part of) or determine objects that stand in a close relationship, such as a parthood relation. In [98], this was generalized in the form of inter-identifier links, a notion that subsumes aspects of the typing and the collection layers. Given larger numbers of objects, it may be desirable to have this type of information available much more rapidly than domain metadata.

4.2.4 *Abstract data type definitions*

As mentioned above, the preservation-centric model relies on a notion of a Persistent Entity as an abstract data type, a pragmatic interpretation of the more formal definition. In the full model, this Persistent Entity ADT will be present as an agglomeration of functionality from different layers, resembling a facade for some of the full stack functionality. The ADT can only be provided through preserving PID systems, i.e., criteria 1-3 and 6 must be met.

The Persistent Entity ADT consists of the following atomic operations; naturally, the same operations apply to Digital Objects. A description with slightly different scope can also be found in [98], which does however not fully reflect the distinction between layers described in this chapter.

CREATE INSTANCE: This operation's parameters are an identifier name, resource location and PID record metadata. The operation fails if the identifier violates criterion 2, i.e., if the identifier is already globally resolvable.

SET RESOURCE LOCATION: The central idea of PIDs to establish a layer of redirection (cf. figure 6) relies on an operation to modify the location of a resource after a PID has been assigned to it (criterion 1). As also described in [98], the operation should confirm that the resource at the new location is identical to the one at the old location, to prevent violation of criterion 6. This may for example be done by verifying a checksum in the PID record, provided on initial assignment. Since the resource at the old location may already be unavailable at this point, verification may however be not be possible at all times. Even with a checksum present, policies are required that make sure the process is quality-controlled — if there is a procedure in place that ignores the checksum verification, the criterion will be easily violated.

GET RESOURCE LOCATION: In simple terms, this operation returns the last parameter to the 'set resource location' operation or the

'create instance' operation. Note that because the resource is not part of the Persistent Entity, it is not possible to offer a full 'get resource' operation. Whether the resource still resides at the location is intentionally not a concern of the Persistent Entity.

SET PID RECORD: The information contained in the PID record is subject to possible change in many situations. As explained earlier, the scope of metadata cannot be restricted at this point. It may be worthwhile for an implementation to differentiate between protected or immutable metadata and mutable metadata. Typical protected metadata includes for example the checksum, the date of object creation and original creator. If the metadata contains relations to other objects, these should be Persistent Entities, so that the operation can verify that the given identifiers are actually resolvable. Otherwise, the resulting graph will be broken. The actual availability of the target resource may be ignored if the given object is a proper Persistent Entity ensuring criterion 3.

GET PID RECORD: This operation simply returns the PID record contents set earlier.

Another possible and typical operation is to remove the instance. However, this would cause a violation of criterion 3, resulting in the identifier becoming unresolvable, and thus it is not part of the ADT definition.

Finally, to make the ADT model complete, it must be possible to retrieve actual PE instances. This can be described by a rather simplistic resolution service ADT with only one operation, as also described in [98]:

RESOLVE IDENTIFIER: Given an identifier, this operation either returns an instance that bears the given identifier or fails if no instance with the given identifier exists [98]. Here, it is beneficial to employ a PID system conforming to criterion 5, but this is not strictly required.

As mentioned earlier, two abstract operations can be distinguished: resource resolution and record resolution. The Persistent Entity ADT provides the record resolution operation. The resource resolution operation is a bit more complex as it must combine the resolver's 'resolve identifier' operation, the instance's 'get resource location' operation, and some additional operation at the secondary preservation level to actually retrieve the data. This transcends the scope of our ADT definitions, and thus it is up to actual implementations to fully enable this operation.

4.3 TYPING LAYER

The formal model and the persistency layer address fundamental questions on terminology, persistency, system requirements and trust, but they do not describe the actual contents of PID records available through the dedicated Persistent Entity ADT operation. A central question put towards PID systems by user applications concerns the nature of the objects if only their identifier is available. With PID records, there is a chance to provide more information without having to retrieve the object, and part of this information should help in determining the *type* of object at hand. The goal of the typing layer is to address this typing problem. Perhaps unsurprisingly, the means to achieve a suitable solution is the application of subtyping or inheritance principles. In [45], the notion of types of digital objects was introduced along with the notion of a type registry, which are further extended in the following.

Earlier, PID records were defined in terms of a mapping from keys to values, but there were no further details on the nature of either keys or values. The typing layer refines this model so that there is additional information available about the characteristics of each value such as its value type.

This section adheres to the following internal methodology. Two main practical interpretations for the use of types in relation to persistent identifiers are the typing of objects and the typing of PID records. In terms of organizing the contents of PID records, two practical organizational models are possible that differ in complexity and relate to different interpretations of properties, types and profiles. These practical observations shape a formal terminology based on registered properties and profiles. For practical adoption, it is also required to define the elemental roles of the type registry and a type governance process.

4.3.1 *Typing of identified objects*

When talking about types, the most elemental use case is concerned with the type of the identified object. Does it represent a scientific data file, a metadata record, a digital document? And what are the defining characteristics for classifying it as a specific type? Does its contents, for example, follow a particular data format?

If the object bears a PID and the PID is associated with a PID record, then the simplest solution for enabling quick determination of the object type and potentially preserving this type beyond the object's lifespan is to include essential information in the PID record. This information may offer some fundamental categorization (data object, metadata object, document and so on). For some use cases such information is already sufficient. If, for example, an automated

operation merely needs to filter out all objects that are data objects from a diffuse collection, a simple categorization may suffice. Another service may discover that an object that was deleted a while ago was a data object, but its related metadata record may still be available at another location.

However, in some real-world scenarios the capabilities of a simple classification scheme are quickly exceeded as already indicated in section 3.6. For example, a service may rely on a distinction between data and metadata objects that is not easily determinable through a single generic categorization. Some services may only work on data objects that follow much more specialized characteristics, and require conversion for others or reject them as input. Such a complex notion of the "type of an object" may thus not be determinable from a single entry in the PID record, but require a specific pattern of multiple values being present. A type description is the formalization of such a pattern, and determining the type of an object comes down to querying whether its PID record conforms to the formalized pattern. In contrast to a simple categorization approach, this provides more flexibility on the typing because it is not any more the sole task of the PID record generator to make a best guess, which is in principle a hard or impossible task to perform due to the generator's very limited knowledge about a priori unknown but possible future use cases. Rather, the stated goal of a typing mechanism should be to enable classification actions by the specific actors examining the PID record without contact to the record generator.

The typing of identified objects is also important if the scope of identification is extended beyond data objects, to include e.g. scholarly articles (e.g. DOI), people (e.g. ORCID), organizations (e.g. ISNI), samples (e.g. IGSN), geospatial features and so on. Currently, there are usually different and unconnected identifier systems used for the different entities which allows for decisions based on the particular naming scheme. If there is a move towards a more comprehensive identification system in the future, then the naming scheme cannot be used anymore to determine the fundamental type of an unknown entity purely by looking at its identifier name.

4.3.2 *Typing of PID record entries*

Complex typing of objects may work through creating and detecting specific patterns in PID records. Until now, the contents of a PID record was treated as a black box, with the single exception that it usually contains a link to the location of a bit-stream and often some form of metadata. The structure of PID records must therefore be examined in more detail. When trying to formalize patterns in the PID records so that they can serve categorization purposes, both keys and values need to be considered. The following basic terminology

on the PID record entries has proved useful in the discussions within RDA as well and is similar to efforts made in the early days of the DOI System that led to the definition of "DOI Application Profiles" [25, chapter 5.5].

A PID record consists of *properties*, each encoded as a key-value pair. The property key may not be unique for a single PID record; multiple values for the same key may be present. The property key should be a persistent identifier which can be resolved to a persistent property definition record or *registered property* in a registry. The property definition is identified through a PID and specifies at least the *property name* meaningful to humans and the *property range* (value type) which should be a PID as well. Possible optional entries (as are used by the Type Registry prototype) include: information about the creator of the property, a description text explaining the usage of the property intended by the creator and arity constraints. Without the property definition, a PID record relying on it may become incomprehensible. The property definition must therefore be subject to digital preservation, meaning that both its identifier and contents must be preserved for at least as long as the PID records employing them. Since the entity preserving the property definitions cannot know all PID records in existence, it has to plan for long-term preservation of the definitions. The PID system that redirects to registry entries must be a preserving PID system.

Typical property ranges include "string", "boolean", "integer", "date", "time" and "geolocation", similar to the common primitive data types of programming languages. A special property range is "identifier". The difference between identifier and string ranges is that an identifier conveys an inherent promise that the property value is resolvable. This is an assurance given by the record's creator, which can only be verified by an actual resolution action. Using a preserving PID system may increase trust.

In an alternative model, the property range is part of each PID record entry: A PID record consists of properties, encoded as key-value pairs, where the value consists of a value type and value data. Property keys may be persistent identifiers, which can be resolved to property definition records as stated above but without a unified property range. Doing so will however lead to insular non-interoperable adoption and conflict with the stated goal to support a strong governance mechanism. This is even more the case if properties are not registered, so that provenance information is less easily obtainable and verifiability suffers.

4.3.2.1 *Structure of typed information*

Out of practical discussions came two concrete models for structuring typed information: the property-type-profile model and the simpler type-profile model. These models reflect the different mindsets

and disciplinary backgrounds the RDA working group participants came from. The two competing but not necessarily incompatible views provide important insight into the practical expectations and requirements. They should be understood before a conceptual unification can be made that takes some of the practical considerations into account. Note that as a consequence of the actual practical discussions, the internal terminology in the next paragraphs is slightly inconsistent; particularly, the terms *type* and *profile* are heavily overloaded. Both models are also described in the working group's final report, also to appear as an RDA recommendation [96]; more context on the practical discussions can be found there.

In the **property-type-profile model**, a type consists of a number of properties, which are subdivided into mandatory and optional and referenced by their respective PIDs. Every type is registered in the type registry and bears a PID along with a description text and additional information. If a PID record contains all mandatory properties of a type, it conforms to the type. A profile consists of several types and thus offers a second aggregation level. Types can also be classified as mandatory or optional with regard to a specific profile. A PID record conforms to a profile if it provides all mandatory properties of any constituent mandatory type. The complementary view is true as well: If a service answering profile conformance queries replies with a positive result and if the caller trusts the service's general assessments, the caller can be sure that a certain set of properties is available from a given record without having to parse the record in detail.

There are also two minor distinctions between profile implementations. A profile may be registered so that it bears a PID and is globally discoverable. If this is not the case, a profile is only "local", e.g. only known to a distinct community or implemented service. It may even be understood and emulated as a simple list of type identifiers. This lowers the barrier for using profiles, but such profiles are less shareable and re-useable. Registered profiles, in contrast, can be used by third parties, such as other disciplines or community e-infrastructures, and provide better accountability through the additional information available from the registry. Such can be useful for example to dispel doubts whether a profile is fit for the purposes of a specific community if it was not designed by them.

The motivation to employ profiles may also be of a non-technical nature, hinting at the desire for a practical governance mechanism in some form. One motivation for using profiles, particularly for registering them globally, is to preclude the proliferation of types that may occur if e.g. a community intends to reuse a predefined type but actually requires only one additional property to be present. The proliferation can however not be prevented completely, as there are also semantic differences to consider even if two types list identical

properties. Another motivation is the frequently expressed wish of communities such as EUDAT, CLARIN or DARIAH to have exactly this kind of profiles mechanism in order to mark PID records as originating in the particular community and thus being able to enforce some form of internal standardization and external branding.

The **property-profile model** is simpler insofar as it offers only one aggregation level. The overall concept can be traced back to linguistic communities (DARIAH, CLARIN) with considerable history in using PIDs and internal use by CNRI in its original Handle System concepts. A different terminology has been used traditionally in these domains: The particular understanding is that the elements of PID records are typed, and the types may be registered in the type registry. To provide some form of aggregation, also in view of usage by particular communities that will work with a distinct set of types, the notion of a profile is introduced, sometimes even called a "community profile". Given this background, the model may be termed a "type-profile" model, but this may cause even more confusion given the previously defined terminology with properties as the atomic elements of a PID record. The complexity of an additional layer as in the property-type-profile model was perceived as being too cumbersome and unnecessary, thus the preference for a model with only one aggregation layer.

In any of the models, a service may require the presence of a specific type or profile on PID records it works with, and reject those that do not conform. Aside from the potential for solving interoperability issues, this is a functionality often mentioned as being valuable in practical implementations.

4.3.2.2 *The generalized model*

The generalized model structures the PID records and provides a mechanism that covers what has been called typing in the described practical scenarios. The model includes two distinct type hierarchies, one for the types of *property* values in the PID records and one to aggregate the properties themselves into structured entities that can be the target of conformance queries. These are termed *profiles* in view of the discussions of the previous section. Profiles are typically assembled according to criteria that are meaningful for application use cases.

The fundamental elements of the property type hierarchy are registered properties. A registered property is a 3-tuple of a name, a property identifier (which is a PID) and a value type. The property name is a simple string, the value type is referenced through its PID. Instances of registered properties are expressed through the individual key-value pairs of actual PID records. The elemental properties can be further organized according to one of two models: subtyping and aggregation.

The key concept behind subtyping is given by Liskov and Wing [57]: A subtype preserves the behavior of a supertype's methods, its invariants and history properties. Invariants are properties that are true for all object states, while history properties are true for all sequences of states. Preserving a type's behavior means that its method pre- and postconditions must remain intact through any subtyping.

A profile can only be a subtype of properties, and not vice versa, because otherwise the behavior of conformance queries is inconsistent: A record may conform to a property (carrying its single value), but not a profile (carrying all of its mandatory values). When understanding a profile as a subtype of several registered properties, the behavior of the supertype must be preserved, meaning that the profile must provide the same methods. Profiles and properties are however different in usage and it is unclear whether all methods can be easily provided. Two possible solutions come to mind. The naive solution would be to parametrize the methods of each registered property, such as "get value" and "set value", so that they carry the semantics of a specific PID record element like "checksum" in their signature ("get checksum value"). This seems rather cumbersome and hardly scalable. Alternatively, a type hierarchy can be constructed where properties and profiles are subtypes of a common supertype such as "registered structural object". This will however not solve the problem because then profiles are not a subtype of properties and the intention to employ subtyping seems inadequately matched.

In the aggregation model, a set interpretation is used which resembles the state extension model explained in [57]. A PID record then consists of property instances. Each property instance is a 2-tuple of a registered property PID (the key) and a value. Many property instances can be associated with one registered property, resembling the concept of instances of a type or class. A registered property is a 3-tuple of a PID, a name and a value type. A registered profile is a 3-tuple of a PID, a name and a set of registered properties. It aggregates multiple registered properties and through them the property instances of a given PID record. A registered property is not an instance of a registered profile, and the registered profile is consequently not a metaclass. Instead, the nature of this relationship is that of an aggregation where the aggregated elements maintain their identity and existence independent from the aggregation. Definitions for properties and profiles are registered in a type registry, an activity through which they receive their respective PIDs.

To use profiles to construct new, more complex profiles, their respective sets of properties are merged. Special precaution must be taken to store information about which profiles were used, because this will not be inferable from the result. If the intersection between the profiles is not empty, i.e. if there are properties of same name,

ENTITY	MANDATORY ELEMENTS
Registered property	property identifier; property name (label); value type identifier
Registered profile	profile identifier; set of property identifiers
Registered value type	value type identifier; value type name (label)

Table 1: Minimal data model for the type registry. A registered property record may also specify a cardinality (cf. section 5.2)

it must be verified that they express the same semantics. This mechanism of constructing new profiles is essential because it addresses the need to aggregate elements in a hierarchy as expressed in the property-type-profile model.

In conclusion, the subtyping metaphor may be more familiar to past users but could also cause a lot of confusion. Is it always intuitive to use a profile instance in place of a single property instance? With respect to the described issue of methods and inheritance, this is questionable, and thus the typing layer relies on the aggregation interpretation for constructing registered properties and hierarchies of profiles. Because the ultimate intent of the typing layer is to type the object behind the PIDs, using the notion of typing appears reasonable even if subtyping is not applicable in the strictest sense. Also note that there is no theoretical limit on the number of profiles and hierarchical depth one can define; the limitation of the exemplary models to one or two levels is pragmatic but completely arbitrary.

In contrast to the property hierarchy, the value type hierarchy conforms well to the subtyping requirements. It starts from types such as “string”, “boolean”, “integer” and similar. A subtype of particular importance for thematic linking between objects is the “identifier” subtype of “string”. One can construct further subtypes of “identifier” that are defined by constraints on the identifier’s record, e.g. conformance of the record to a specific profile. This is all in line with typical examples of type hierarchies as are also given by Liskov and Wing [57]. The type registry will store value type identifiers, names for them (that effectively constitute a simple controlled vocabulary) and possibly further restrictions on the syntax of conforming values, which remains undefined at this point.

Table 1 presents an overview of the type registry data model.

4.3.2.3 *Conformance queries*

Conformance queries constitute the main mechanism employed to determine the type of an identified object. Colloquially, a conformance query will tell whether a given PID record conforms to a

given registered profile; conformance towards single properties is also possible, but less useful. The conformance method is an integral part of the typing layer and its top interface. But how should profiles be designed so that the conformance queries are practically useful?

There are two possible models for conformance query semantics:

WEAK CONFORMANCE: A PID record conforms to a profile if it provides all property keys specified in the profile and if there is a value associated with each of these keys.

STRONG CONFORMANCE: A PID record conforms to a profile if it weakly conforms to it and if each value is valid in the range of the respective property.

A typing layer implementation must be able to confirm weak conformance but may be unable to assess strong conformance, because the range of valid values may be indeterminable for it without consulting the third party that defined the respective property range. This is particularly true for more complex subtypes of "string"; it may already occur for example for simple dates due to the multitude of possible encodings.

A profile may be designed to contain all essential properties of a specific type of object. In simple applications with clearly defined object categories and no or insignificant external factors such as PIDs from third parties, this may suffice. In more complex scenarios, the variety of use cases and, in consequence, the variety of actions to be performed on objects may quickly lead to the development of a profile hierarchy, where each profile only covers a certain aspect of an object type. The definition of a profile thus depends on both the type of object and particular use cases. Conformance queries are then essential to filter the currently valid actions from the set of all actions possible in the application.

4.3.2.4 *Inter-identifier links*

Functionality valuable to higher layers is to establish typed links – binary relations – between identified objects, close to the notion of Linked Data. In [98], inter-identifier links were defined to be a part of general metadata at the persistency layer. As also described there, a more sophisticated model will extend this to the notion of an *identifier graph*. With a dedicated typing layer, linking functionality would be offered as an implementation of the typing facilities by subtyping the "identifier" value type. Such subtypes then classify different links; the result is a graph of identified objects, where the PIDs (representing identified objects) are the nodes and the links between them are the edges. Section 7.1 provides a more in-depth comparison and possible convergence of these concepts with Linked Data.

In terms of the typing layer, some aspects must already be mentioned at this point however. The subtypes of the "identifier" value type act as labels on the edges. Because value types are bound to registered properties and these can be used in multiple profiles, the edges are always directed. The object at the source of such a directed edge is not entirely constrained, because the record may conform to more than one profile specifying the property. If a PID record is extended with more properties, the number of conforming profiles may increase, and thus the type of object at the source of the link is not entirely fixed. The type of object at its target may however be specified in the registered property record and verified by the typing layer. The property record may also include edge weights or other annotation information valid for all respective links; it is not possible to specify edge weights on individual links unless the PID records are enhanced accordingly.

4.3.3 *The type registry and type governance*

As described earlier, value types, properties and profile descriptions should be registered so that they can be globally referenced and retrieved by PIDs. The original motivation goes even further: Object types should be registered so they can be disambiguated. Domain data object type examples are "ESM output data", "Temperature measurement time series data" or "Remote sensing dataset". Registered high-level profiles, possibly as agglomerations of several other more specific profiles, may effectively stand in for such object types. Accessing the profile definition through the type registry will then help to verify the object's fitness for purpose through conformance queries. The type registry should also provide some metadata on the profiles and properties, such as information on the creator, currently responsible party and intended purpose. Initially, it may provide a small set of approved value types, and even some generic properties. But who registers new value types, profiles and properties, who is responsible for maintaining them, and may additional effort to build a central governance structure be justified?

Registration of new value types, properties and profiles may be done by users who generate PID records. This may include users from a community e-science infrastructure, libraries, publishers and similar. It is also possible that larger cross-disciplinary initiatives such as RDA contribute to the registry. The type registry may be set up in a distributed manner, where each community or other entity interested runs their own instance and an overarching service takes care of global resolution (cf. criterion 4). The question then is how the multitude of registered properties can form a coherent whole when they are principally defined and registered independent from each

other, e.g. by different communities. Some properties or value types may be equal or overlapping in definition.

Two approaches can alleviate the problem: mappings and central governance. Colloquially, these may be described as "free market" and "regulated market" models. In the mapping approach, mappings between between individual value types and properties (or even profiles) defined at different type registry instances may enable interoperability if exposed at each of the mapped registries. This is similar to semantic mapping approaches known from the Semantic Web (see e.g. [37, section 6.3.2]). The fundamental problem of this approach is that the original creators of the types may not feel responsible for defining such mappings since usage of properties within a single adequately coherent community can work well without mappings. It is only when use cases span communities (i.e., type registry instances) that mappings are required. Those actors interested in enabling such use cases may not have the deep knowledge about the types the original creators have and will thus be more challenged to make correct mappings. The advantage of such an approach can be far easier adoption because there will be no regulations on defining properties or profiles.

The alternative solution may be a central governing authority that enforces business processes for all newly registered value types and properties, which ultimately require some form of approval. This may slow down adoption, but will at least increase the likelihood that some of the value types, properties and profiles are shared and re-used. To achieve this, it may be worthwhile to provide end-user search facilities and overarching documentation about the ecosystem of types already registered. Profiles may also be split to reach agreement on smaller aspects. An open question is who will provide the resources for maintaining the central governance structure.

Aside from the two approaches described above, a variety of mixed cases may emerge. Ultimately, however, the type registry lies in a conflict zone as it strives to fulfill two fundamentally conflicting requirements: first, to provide some level of unification and interoperability among the types, so that potentially pieces of information are interpretable by third parties, and second, to ensure that every community can precisely define the types that match their needs best because they are usually best suited to judge the fitness for their use cases. Interoperability may eventually be better achievable at a higher layer, for instance through Semantic Web applications mediating between types. This is however not a core concern of the type registry which first and foremost aims to provide persistent identifiers and maintain the type definitions over longer time spans and only secondarily addresses content issues. The justification for establishing a centrally governed type registry system then stems

from the estimation that the anchoring of types is required to achieve interoperability as a depending goal.

The type registry cannot assume knowledge about the actual use of the properties and profiles that are registered with it, because it cannot know the entirety of PID records. At a technical level, the type registry must therefore use a preserving PID system: It must ensure that the definitions are provided over long time-spans, and even if some definitions are deleted, the PIDs should be resolvable to a notification marker. It therefore depends on a lower preservation and foundation layer just as the PID records that make use of the types. The underlying implementations may be completely separate, e.g. if different PID systems are used. The type registry should therefore be seen as a service external to a typing layer implementation as it is also in principle exchangeable (also see later in figure 7). It must also adhere to high standards on Quality of Service and preferably be established as a highly distributed, highly available system. On the other hand, its records may be assumed to be static over significant time scales, and therefore local caching at typing layer implementations may work well.

Finally, the type registry should be agnostic towards the entities it registers, but since the typing layer relies on an understanding of registered value types, properties and profiles, these in fact make up some first elemental types. If all profiles are known, e.g. because there is only one type registry instance designed as a coherent distributed database, the typing layer could offer methods such as listing all profiles a PID record conforms to.

The typing layer presented here in this form is independent from the particular implementations of the lower layers. It relies on the persistency layer to continually resolve PIDs and return the PID record contents. The criteria of the persistency layer can ensure that this information is reliably available over time and trustworthy. The type registry forms an external dependency that in consequence also depends on its own persistency layer.

4.4 COLLECTION LAYER

Collections[2] of digital objects provide support for steps in a number of use cases presented earlier. The overall purpose of collections in this sense can be summarized as follows:

- Collections enable machine-orchestrated operations across multiple objects.
- Collections can make implicit parthood relationships between digital objects explicit.

2 The term *collection* is in common use in communities and projects employing PIDs.

- Collections provide abstraction: Depending on the context, either an identifiable whole may be addressed or, at other times, individual objects. The composition of the whole may even change over time, but through the abstraction, an agent referring to the whole can remain oblivious to such internal changes.

The definition of collections, their composition and defining properties can be explained by a brief formal model. This also includes a description of a general process for the creation of collections and possible changes over time. Some essential characteristics can help to classify different usage scenarios for collections. Some of the usage scenarios are also constrained by the criteria of the persistency layer. Finally, collections are actionable constructs, and therefore operations on them can be defined, following similar intentions as with the ADT interpretation of the Persistent Entity. In [97], a comparable notion of actionable collections as ADT instances was explained with particular emphasis on a practical implementation with Handle System identifier records, however lacking the more fundamental aspects including a life cycle approach to collections as described in the following.

4.4.1 *Definitions and terminology*

In the simplest view, a collection is a set: an object which consists of a number of elements. This has also been indicated by Kahn and Wilensky in [45] by describing entities such as a "set-of-handles" and a "set-of-digital-objects". Every collection element has its own identity and exists independent from the collection. Only collections that consist of digital objects are considered here; thus, the space of collection elements is the space of digital objects, and every collection element bears a PID.

The simple view is however not adequate to describe the change of collections over time. An adequate model is the notion of a particular state of a collection at a given point in time. Then, a collection consists of a sequence of collection states, and each collection state is a set (or multiset, see below) of digital objects. In reality, such a sequence will be finite and consist of a number of states growing over time. A collection with a single state is called a static collection. A collection with more than one state is called a dynamic collection. The first sequence element always describes the initial set of objects in the collection when it was formally created. A collection with an empty state sequence is called an empty collection and of little practical value except as a transitional object before its first state is defined. When talking about the elements or items of a collection colloquially, a specific collection state is usually meant.

In some cases, we would like a collection to be ordered, meaning that the elements of collection states should be ordered. This is useful, for instance, to review the order in which elements were added to the collection, or to encode an ordered sequence of versioned digital objects. The ordering can be described by stating that there is an order relation defined on the cross-product of the element space. The order may be partial or — perhaps more typical in practice — total. An ordered collection then consists of a sequence of collection states and an associated order. We can even imagine a more complex situation where there is a sequence of orders, each element of which is associated with at least one collection state, and each collection state is associated with exactly one order. Such a situation may however be less useful to study since many use cases require that the order remains constant over time so that changes between collection states can be described in terms of e.g where new objects were inserted.

Introducing ordering to collections also quickly leads to the notion of having a collection state with several identical elements at different positions. This can be easily expressed by generalizing the notion of a collection state to be a multiset of digital objects.

4.4.2 *The collection process*

A collection is subject to a process with several stages over its lifetime, ranging from defining the collection properties, its initial members, to extending it and possibly fixation. The collection life cycle can be aligned to the data life cycle models presented in section 2.2.

4.4.2.1 *Defining the collection properties*

A principal property of a collection is whether its elements are **ordered or unordered**. Although it is theoretically possible to transform an unordered collection into an ordered one after elements have already been added, the additional information required to order the elements in hindsight may be impossible to attain. This property is thus such a fundamental one that it is best defined at a very early stage before adding actual collection elements. The opposite situation is much easier to cover: An ordered collection may be turned into an unordered one if we define unordered collections on multisets.

A collection may be defined according to **thematic criteria** that describe which elements belong to it and how they relate to each other. Note that this is strictly optional: To enable the review of the thematic criteria by third parties, they must be encoded as part of the collection's metadata. This may not be done for every collection, possibly due to reasons such as the costs involved in formally encoding the criteria or because the collection is created as the output of an automated process that does not have the necessary information or sophistication.

Defining the essential properties of a collection is usually a task taken at a stage in the data life cycle just prior to when data are shared with third parties for the first time, e.g. just at the entrance boundary to the shared research domain as defined by Treloar and Harboe-Ree [90] or at the receive or appraisal stage of the DCC model.

4.4.2.2 *Determining the initial set of elements*

At this stage, the initial set of collection elements is compiled, the 'proper' collection is created, i.e., the first collection state. If an element does not bear a PID, it must be given one as part of the joining process. The collection may receive an individual PID, however this is not strictly required as seen further below.

A principal decision to take at the collection creation stage is whether the **administrative responsibility** of member objects changes. The administrative responsibility for the object determines who maintains its identification relation and the object contents. If responsibility is transferred to the collection creator, preservation of the collection and its elements is certainly easier as it does not cross administrative boundaries. If responsibility is not transferred, the collection is *virtual*: Its elements may originate from administrative domains different from both the collection and each other. This leads to a possibly difficult challenge in digitally preserving the collection and all its contents.

In both cases of administrative responsibility, several nuances in the strictness of preservation can be distinguished. At the strictest level, all member objects are preserved, requiring a preserving PID system. At the intermediate level, only the PIDs are required to remain resolvable to at least some essential information (requiring criterion 3 to be fulfilled). At the lowest level, there are no constraints on the preservation of PIDs or resources.

In any case, the collection object must be preserved, which most notably includes its metadata that describe which elements belong to it by referencing their PIDs. The collection object is therefore a Persistent Entity: By establishing relations to the Persistent Entities of its member resources, a collection can remain intact even if some of the member resources are gone. In terms of the data life cycle, the task of determining the initial set of elements is made at an ingest or publish stage when sharing data.

4.4.2.3 *Extending a dynamic collection*

If elements are added to the collection after its initial creation stage, the collection becomes a dynamic collection with a sequence of collection states. Elements that are added must observe the same conditions that are valid for the initial elements; they receive PIDs and are subject to the same preservation rules. Depending on the

implementation, it may be impossible to add elements to a collection that was created in a static fashion; however, this is not a hard conceptual restriction.

Some procedures may have to be performed after extending a collection. For example, if there is a mechanism defined that provides a checksum over all elements, the checksum may have to be updated. If the collection bears a PID, the corresponding system will not be able to ensure criterion 6, which is however quite intentionally.

Regarding the data life cycle, extending a dynamic collection may be part of the iterative cycle of reusing and repurposing data, but also occur at a general publication stage for example in the case of continuous time series.

4.4.2.4 *Collection fixation*

Eventually, a dynamic collection may undergo a specific fixation stage and no longer receive additional elements. This process may involve for example to set administrative flags in the collection properties that prevent future extensions or to calculate a final checksum. In terms of the data life cycle, fixation will happen at a preservation stage, after which no changes are possible anymore.

4.4.3 *Fundamental structural criteria for collections*

A collection implementation can offer functionality at different levels of sophistication. Such essential functionality can be characterized through a set of criteria. The criteria can be used to classify different implementations and, accordingly, their fitness for purpose in view of exemplary use cases. In the end, conformance to specific criteria also determines which operations are possible on collections and their members. The criteria do not depend on particular process stages as discussed above; an implementation should be evaluated whether it matches a specific criterion across all stages. As described in [97], such criteria can also be interpreted as fixed design requirements for a particular solution with respect to a limited set of use cases. However, not every solution will have to conform to all criteria at all times as not every driver use case will require this.

The criteria are equivalent to the corresponding requirements in [97, p. 196-197] and reformulated to better match the notion of collections presented here; the last criterion is formulated significantly more precisely.

Collection Criterion 1 *Multiple membership: Every digital object may be an element of two or more distinct collections simultaneously.*

If the implicit relation between collections and member elements is assumed to be injective, this criterion remains unmatched. If the

criterion is fulfilled, the relation is not injective and thus the collection states may overlap, i.e., element sets of two distinct collections can have common elements. If the criterion is not matched, it can be safely assumed that the intersection of any two distinct element sets is empty.

The notion of simultaneous membership is important with respect to dynamic collections. Imagine a case where there are two collections and a digital object that is member of each of them consecutively, but not simultaneously. Such a situation may be supported by an implementation that does not provide simultaneous membership in both collections. The criterion thus includes the notion of simultaneous membership.

Collection Criterion 2 *Nesting: Any collection can be nested as an element in any other collection.*

This has only been hinted at in the base model, but it is a valid practical requirement and has large impact on the level of sophistication possible when serving specific use cases. Being able to construct collections that have other collections as their members — possibly including recursion — provides a powerful tool, but also requires a higher level of sophistication from the implementation and possibly raises scalability and operation complexity concerns. Fulfilling the nesting criterion is a necessary condition to construct trees of objects. An implementation may choose to enforce a policy that restricts the nesting depth.

Collection Criterion 3 *Dedicated PID: Every collection bears a dedicated PID identifying the whole object apart from its members.*

It may seem surprising at first, but it is not strictly required that a collection bears a dedicated PID. An implementation may not provide such *head PIDs* and still offer the facilities to bind objects together and operate on the whole, for instance through distinct metadata records. For the criterion to be fulfilled, every collection must bear at least one head PID; possibly more (see equation 1). Having a head PID available for every collection also makes it easier for collection operations to refer to them.

Collection Criterion 4 *Full navigability: Given an element PID, it is possible to navigate to the collection object and to any other element of the same collection.*

There is also the notion of simple navigability, expressed as being able to get all elements PID for a given collection. Simple navigability is an essential requirement for any notion of a collection, because it is a defining feature. Note that due to the simple navigability requirement, being able to navigate to the collection object is already

sufficient to match this criterion, because by going through the collection entity first, all siblings can be reached. Note that it is not strictly required to fulfill collection criterion 3 at the same time, although it may be quite useful.

4.4.4 *Common operations on collections*

The notion of a collection is very close to the common principles of typical abstract data types (ADTs), originally presented by Liskov and Zilles [56] and nowadays an integral part of typical higher-level programming languages or their standard libraries. In this restricted view, the notion of an identifier is also similar enough to that of a pointer to a memory location. Common ADTs include lists, sets, multisets or bags, trees, graphs and maps, and the notion of a Collection is particularly familiar to users of the Java programming language. Unsurprisingly, common operations on collections are strongly related to the typical operations possible on the aforementioned ADTs and include adding or inserting an element, removing elements and retrieving all elements of a collection or iterating over them. If collection criterion 4 is fulfilled, we may also retrieve one or several collection objects for a given element and iterate over all sibling elements. If this criterion is not fulfilled, it is not possible to navigate to the collection object; however, it may still be possible to navigate to other elements (for instance if the collection is a linked list).

4.5 DISCUSSION

The framework presented in this chapter transcends the mere redirection layer based view of persistent identification (see section 2.4.2) by integrating PID records as an essential component and enabling higher-level services based on them. For scenarios where the full layer model is perceived to be too complex because some of the more sophisticated facilities such as typing and collection building are not used, an alternative view is possible by following the preservation-centric model that emphasizes the Persistent Entity abstract data type as an interaction tool and through it a distinction between two levels of preservation.

The typing layer provides an essential component to triage unknown objects and making decisions independent from a particular object origin, location or PID system. Through the typing profiles mechanism, such objects may in fact be typed in several different ways simultaneously, for instance to reflect different interpretation and requirements from individual communities. With typed and interoperable information available from PID records, it is not necessary anymore to include meaningful elements in PID names, a

practice that can lead to serious issues when considering long-term preservation perspectives. Inter-identifier links, but also collection information, gather elemental context information about scientific data, and because they are preserved at the primary level, they can persist beyond the existence of the data objects involved.

The choice of layers is of course not entirely fixed and alternatives are possible. The most important alternative concerns the use of inter-identifier links. Conceptually, one may introduce a specific *linking layer* between the typing and the collection layers, the purpose of which would be to provide a notion of typed links between resources that are both rapidly accessible and in principle available beyond the resource's lifetime. Linking-related operations include, for example, to verify link stability, i.e. to confirm that all records are still available, and if a type constraint on the target object is defined, this may include verification of the the constraint, using the typing layer's profile definition and target PID record. Such functionality may however also be provided by the typing layer at its top interface because it already contains all required facilities. The three-layer model is based on the estimation that typed links are an application of typing, but not distinct enough to justify a separate layer. In view of Linked Data applications, there may be more justification to introduce this as a separate layer, but so far, it does not seem adequate, also considering ongoing related efforts further described in chapter 7.

An open question is how the equality of two Digital Objects or Persistent Entities is defined. The persistency layer describes that PIDs are an integral part of both Persistent Entities and Digital Objects, resulting in the PID becoming an essential part of their identity: if a Digital Object is created with the intention to use several PIDs for the same data and PID record, this will result in several distinct Digital Objects. In the formal model, this is necessary to adequately describe situations where there are e.g. 'default' or 'empty' PID records associated with a larger number of PIDs or where the same data may not only receive multiple identifiers but where also the records may intentionally differ. This is a quite typical practical scenario for instance if a data object receives additional identifiers when it is copied to other repositories, e.g. within the scope of the EUDAT infrastructure. In contrast, if we consider a model that explicitly describes such events, multiple PIDs for the same data can be described as a sequence associating them to the same data (and Digital Object), possibly leading to an adequate definition of object equality. Future work that aims to describe the equality of Digital Objects in a more formal way beyond looking at the identity of data objects should therefore provide such a more detailed formal model.

A particularly powerful concept emerges from the combination of typed records, typed links, and collections, all preserved as part of the Persistent Entity: the notion of a *PID graph*. The fundamental graph

is constructed with the individual PIDs of objects forming its nodes. The typed information available from their associated records may contain typed references to other PIDs; these form the graph edges. The edges are always directed, and of course the graph may contain cycles and any number of typed edges between a pair of nodes in any direction.

Based on this common understanding, there are two fundamental models of how a graph of PIDs can be further constructed. The difference between the models stems from how collections are integrated in the graph. In the first model, the collections are dismantled and their members included in the graph as nodes, preferably typed accordingly. The collection membership relation is then just an additional type of edge label. This model does not break if an identified object is member of multiple collections (cf. collection criterion 1): the subgraph will simply not be a tree with a single root. In the second model, the collections are seen as atomic objects, each collection forming a node with the membership relations and individual members not represented in the graph. Multiple membership will not be visible.

Such a PID graph has broad applicability considering the use cases from chapter 3. It also strongly relates to the notion of Linked Data, which is further discussed in section 7.1. The view of provenance as a directed acyclic graph (cf. [65]) is well suited for the low-level approach linked PIDs with associated records provide. Such a graph can be constructed by connecting the persistently identified input and output objects of a process with provenance-typed edges, e.g. 'predecessor' and 'successor'. In this form of encoding, the provenance graph does however reside at the lower end of expressivity; it does not facilitate reproducibility of the respective processes, nor does it mandate any form of documentation beyond registering input and output data. A layer-based approach which extends such a basic provenance graph with a rich metadata overlay may enable reproducibility, while keeping the provenance DAG as a fallback option in case the more sophisticated overlay information is lost.

If the model of the Persistent Entity is followed, loss of resources will not cause loss of connectivity in the graph, which could potentially break provenance traces. The graph will also benefit from typing, because typing of its resources (through the typed PID records) will allow agents to distinguish types of nodes from each other, e.g. provenance trace nodes, versioned of object states and data from metadata objects.

While PID systems have traditionally provided PID resolution services and can continue to do so as part of the preservation layer functionality, there may be more advanced resolution services that enable the higher-layer functionality, for instance to offer collection-related methods. Technically, such a resolution service would be

situated between the fundamental PID system services and the resources so that it can access the additional information record and perform e.g. further redirection actions depending on the PID record contents.

5 IMPLEMENTATION CONCEPTS

The framework presented in chapter 4 is shaped by the use cases and characterized by a layered design. To enable the use cases, an implementation is required for all layers together or individually. This chapter presents distinct approaches for implementing each layer, based on the notion that each layer shares an interface with its neighbors which can enable a proper modular service design in practice. The number of possible implementations is potentially larger than can be addressed here, including solutions yet to be designed as part of future activities. Wherever possible, the individual sections discuss whether existing solutions are suitable to implement the different layer services. Popular PID systems fulfill some but not all of the preservation layer criteria presented in chapter 4. In some cases, full conformance may be achieved by additionally providing essential policies and an organizational framework or business model. A typing layer implementation must offer a set of operations, some of which can be provided by a Type Registry prototype[1] under development by CNRI.

5.1 PERSISTENCY LAYER

The best candidates for implementing the persistency layer are existing PID systems. As indicated earlier, these can be evaluated in terms of how they meet the criteria and what steps could be taken to improve their conformance. A good overview on the DOI, Handle, Persistent URL (PURL), Archival Resource Key (ARK) and generic URL-based systems with particular focus on the Earth sciences is also given by Duerr et al. [26].

Table 2 summarizes the conformance results. Overall, the DOI System appears to have the best conformance to the criteria, underlining its position as the leading PID system for scientific results.

At first glance, criterion 1 is met by every system listed in the table. This is not surprising since criterion 1 is a necessary condition for a PID system as it captures the fundamental ability of PIDs to point to resources that can relocate while keeping the same identifier. The Domain Name System does not match this criterion and is not included in the table, but instead briefly discussed further below. Criterion 2 is met by every system under consideration, reflecting the idea that a persistent identifier is implicitly globally unique, because otherwise its practical value is diminishing drastically. Who would

1 See `http://typeregistry.org`, last checked Feb. 27, 2015

PID SYSTEM	CRITERIA 1	2	3	4	5	6
The Handle System	✓	✓	?	✓	?	?
The DOI System	✓	✓	?	✓	✓	✓
URNs with local metadata catalogs	✓	✓	?	-	-	?
The ARK System	✓	✓	✓	?	?	?
PURLs	✓	✓	?	-	-	?

Table 2: Evaluation of common PID systems along the fundamental criteria. The question marks indicate at least non-conformance in the strictest sense, but they also indicate cases where there is either insufficient information available, e.g. due to a lack of explicitly stated policies, or where the policies followed by the PID system maintainers are intentionally left open. The latter cases are discussed in further detail in the text.

use a persistent identifier if objects identified by it can relocate to locations outside its scope?

Only the Archival Resource Key (ARK) System emphasizes policies that meet criterion 3. This reflects the design philosophy of the ARK System, which is specifically designed around the idea to ensure persistent access not only to objects, but also to their metadata. Criteria 4 and 5 target the design and behavior of resolution services; these are examined individually for the systems. Criterion 6 heavily relies on policies; in the strictest sense, an authority is required that has some (perhaps contractual) leverage on individual institutions that maintain resolution services or provide storage facilities to enforce the policies.

The following section discusses each system in more detail and points out those aspects that remain unclear or where other criteria are not met.

5.1.1 *Review of individual PID systems*

The Handle System [45] is constructed through a set of distributed servers with a hierarchical identifier namespace. In the past, there has been a single service at CNRI, called the Global Handle Registry®, which maintains the first leg of the namespace. The sub-namespaces are then served through servers at various institutions. These individual Handle servers will not be able to resolve all identifiers, so that resolution requests usually go through the global server. Criterion 5 is therefore not met. Criterion 3 may be met quite easily because the Handle System provides sophisticated technical facilities to associate PID records with the identifiers. However, the question how these records are kept is a matter of organizational policy, which is not

covered by installing a Handle Server and agreeing to its usage policies. The same applies to criterion 6: The resources identified through the Handle System are outside its scope. If meeting the criterion is desired, a policy is required that determines whether this is the responsibility of the institution maintaining the identifier, the agent requesting an identifier, or a third party providing long-term archival services.

The DOI System [73] shares many features of the Handle System as it is its largest customer and prime application. The institutions issuing DOIs maintain a set of policies which target the persistency of resources over longer time spans and prohibit the uncontrolled exchange of identified resources. There is also a process in place where users can alert the International DOI Foundation of broken DOIs. Typical customers of the DOI System include scientific publishers who have their own interests in maintaining the resources and given the wide adoption of DOIs, peer pressure is likely to ensure that policy violations are treated seriously. In view of these diverse social and even economic factors, criterion 6 is met. Criterion 5 is met by the DOI System because its resolution services work on the second tier of the Handle namespace, and thus all its resolvers are able to resolve any DOI. The exemplary central resolver ensuring that criterion 4 is met is quite well-known (`http://dx.doi.org`).

As explained in section 2.4.1, URNs [71, 63] have been around since the early days of the World Wide Web, yet never reached as much adoption as the Domain Name System and the URLs based on it. Notable current users of URNs are the national and scientific libraries. Such an institution may maintain an individual URN resolution service, while the complete space of URNs, divided into individual namespaces, is controlled by the Internet Assigned Numbers Authority (IANA). A global resolver across all URN namespaces is however missing; as explained by Paskin [73], URNs in the strictest form are merely a specification, but not an implemented infrastructure and accompanying policies. Thus the URN system fails to conform to criterion 4. This also means that criterion 5 cannot be met, because each individual URN resolver only knows its own institutional namespace. Since there is no concept of records associated with URNs, we must consider a solution consisting of a local URN resolver and an institutional metadata repository to assess criterion 3. The core question then is whether these records are kept even if the resources maintained by the institution are gone. This seems quite possible, also given that URN policies (cf. [71, p. 1212]) specifically include the notion that a URN should remain resolvable beyond the lifetime of the resource, but in the end, the policies are only enforced at the institutional level, but not by a larger authority such as IANA. Criterion 3 should therefore be considered to be met only partially. Criterion 6 may be fulfilled, because the URN policies

require that an object should not be exchanged, however it remains unclear as well how this policy can be enforced.

As explained above, the ARK system [52] can meet criterion 3 quite well if one trusts its policies and the mechanisms to enforce them. ARK resolution services are provided by individual institutions, and each institution will be assigned its own namespace by the California Digital Library (CDL) on request, similar to the Handle System. As indicated by Duerr et al. [26], there are concepts for a negotiation mechanism that allows resolvers to contact other ARK providers in case a given identifier does not belong to the particular institution's namespace; however, this has not been implemented so far, and since a global resolver is missing as well, criteria 4 and 5 are not met. There is a policy at least promoted by the CDL that demands that no identifier shall be assigned to a different resource, at least supporting criterion 6; however, the CDL will most likely be unable to enforce such policies due to the informal nature of the ARK collaboration. Whether criterion 6 is met at the same level of quality as for example in the case of the DOI System is therefore open.

To evaluate Persistent URLs [94, 4, 26], it must first be determined how this PID system is defined. There are two broad choices on this. In the first model, PURL identifiers are understood as the full URLs, which includes the location of the resolution service, such as "http://purl.org". The PURL homepage lists several other PURL providers, and thus this is not the only resolution service. If the resolution server name is thus part of the identifier name, the primary resolution system will be the Domain Name System. This results in a contradiction, because a PID system is required to only resolve proper PIDs and this does not cover arbitrary HTTP-URLs as explained in sections 2.4.2 and 4.2. The alternative model, which is also the one shown in the summary table, interprets the PURL system as consisting of a number of mutually independent resolvers. The identifier names will not contain the resolver name. Then, such an identifier will not be globally unique, violating criterion 2, and there will also not be a coherent, single namespace and no central resolution service, violating criteria 4 and 5. As with the other PID systems, criteria 3 and 6 depend on enforcing adequate policies. The official web pages and available literature do not include indications on such policies, thus conformance with criteria 3 and 6 remains open.

A preserving PID system is required to implement a Persistent Entity. None of the systems discussed meets all of the necessary criteria for this, often due to a lack of PID record facilities. Where these are missing, local databases storing metadata may be used instead. The Handle System and the ARK System stand out because they provide a notion of PID records as part of their design philosophies. If a solution for PID records is provided, introducing a set of policies and

establishing an authority that can enforce them should then suffice to implement the required ADT operations. In some cases, however, this may be easier achieved than in others. Particularly for the DOI System, the criteria may be met fairly easily if the policies for the persistency of metadata records in case resources are lost or deleted are clarified.

5.1.2 *Scalability aspects*

The research questions mention a particular aspect regarding scalability in terms of how to deal with huge numbers of PIDs. The pivotal element in answering this question is the ability of PID systems to scale. First, this concerns the aspect of resolution. The Handle and DOI Systems aim to achieve this through a multi-tier architecture with root servers that redirect to individual sub-namespace servers, which again can serve requests through an arbitrary number of mirroring servers (also cf. the Handle protocol specification [87]). Secondly, there must be asymptotic constant-time access to the values of individual PID record keys. This can be ensured for example if PID records are stored in a relational database solution with indices optimized for identifier name and key-based access. In general, a PID system with PID records can be seen as a set of nested hash tables.

5.2 TYPING LAYER

As explained in section 4.3.3, an implementation of a typing layer has an external dependency on a proper type registry. This type registry should be centrally governed and in full control of the type registration process. The type registry must offer methods to register and retrieve properties, value types and profiles. A user concerned with the PID layers only will not want to deal with the type registry directly, and thus, some of its methods should also be available through the typing layer interface as well. The type registry must also offer the most elemental value types as part of its initial state. The interactions between applications, the type registry, the typing layer service implementation and PID systems are shown in figure 7. A typical implementation will realize the interfaces as RESTful [29] web service endpoints.

The interface descriptions presented here are also based on the concepts described in the RDA PID Information Types Working Group recommendation [96].

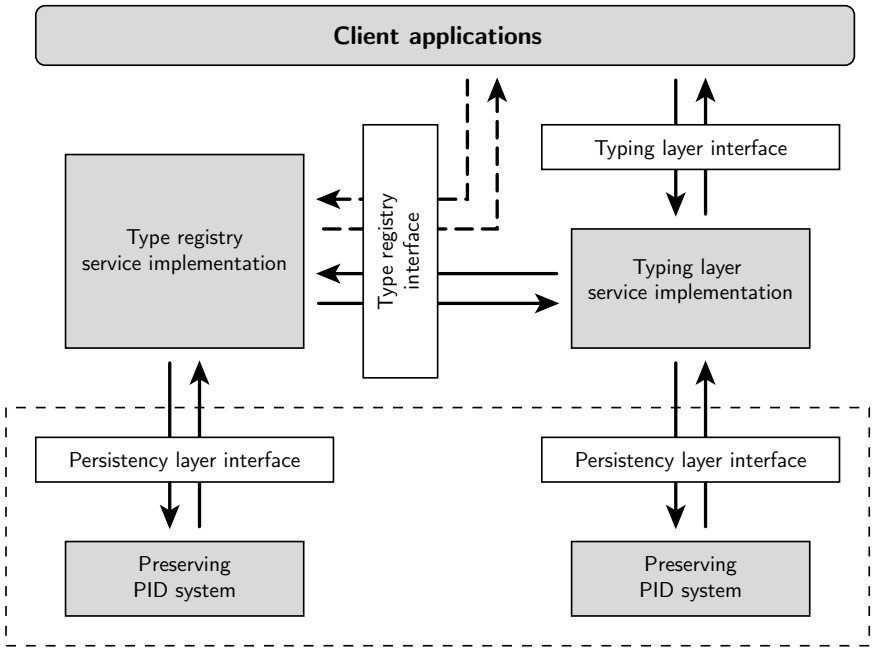

Figure 7: The typing layer relies on both a type registry and a preserving PID system, properly decoupled through separate interfaces. Client applications will usually work with the typing layer service, but may also query the type registry directly. The PID systems and persistency layer interfaces used may be identical or separate.

5.2.1 *Type registry*

The top-level type registry interface must offer a set of mandatory methods, some of which are duplicated in the typing layer. The methods are as follows:

REGISTER PROPERTY: The method registers a property under a new PID given the PID of its value type (the property range) and a name label. Other entries may be included as well, such as a maximum cardinality constraint indicating how often the property may be instantiated in a PID record.

REGISTER VALUE TYPE: Registers a value type under a new PID given a name label and a human-readable description. A type registry may also store additional machine-interpretable constraints.

REGISTER PROFILE: Registers a new profile under a new PID. Two sets of property PIDs must be given (mandatory and optional); each of them may be empty, however, a profile with only empty sets is not very useful. Optionally, a type registry may encourage community adoption by supporting community namespaces as additional parameters for searching profiles. Additional metadata may include a human-readable description

of the intended use of the profile and provenance information such as the profile creator, creation date, and possibly related or predecessor profiles.

MERGE PROFILES (OPTIONAL): As described in section 4.3.2.2, new profiles may be constructed from existing ones by merging their respective property sets. Such a method will accept any number of profiles and create a new profile with a new PID that is based on the union of all respective mandatory and optional property sets. Additional metadata should be provided as well, but most notably, the method must store a reference to the ancestor profile PIDs.

QUERY REGISTERED PROPERTY: Given a property PID, the method returns the name and value type PID of the registered property.

QUERY VALUE TYPE: Given a value type PID, the method returns the definition of the value type.

QUERY REGISTERED PROFILE: Given a profile PID, the method returns the set of mandatory and optional properties contained in the profile, referencing their PIDs and any additional metadata.

In the strictest sense, the PID names assigned by the type registry should adhere to the principles mentioned in section 2.4.3 regarding intelligent identifiers, i.e. they should not indicate the intended meaning or purpose of the respective property, value type or profile. Although it can be argued that the elemental value types are usually quite stable, and the same may be true for properties, the safest solution remains to employ a naming mechanism based on purely random UUIDs.

In contrast to creation, update operations for properties, value types and profiles are rather difficult. The requirement that a type registry conforms to the fundamental PID system criterion 6 prevents any updates to the actual definitions of properties, value types and profiles. This is important because the meaning of PID records employing specific properties will change without notice if property definitions are modified. It is impossible to know and consequently update all PID records that use a particular property or value type. On the other hand, the registered properties and profiles are usually the result of community processes, and therefore they are subject to an initial phase of changes particularly during early adoption. The type registry must deal with these conflicting goals as part of its type governance process and may for instance decide to offer preliminary namespaces or meticulously track the version of registered definitions.

5.2.2 *Typing layer service*

A typing layer service implementation offers a top interface to determine the type of entries in a particular PID record and the overall object type derived from this. Since there may be several typing layer implementations, it is useful to define a generic API to query types. This API should offer the following methods, some of which are the same as for the underlying type registry:

QUERY REGISTERED PROPERTY: Same as for the type registry.

QUERY VALUE TYPE: Same as for the type registry.

QUERY REGISTERED PROFILE: Same as for the type registry.

WRITE PROPERTY TO PID RECORD: This method will write or update a single property in a PID record. Its parameters are the target record PID, the property PID and a value. All PIDs must be checked and the method must fail if the property is not registered. If the type registry provides machine-interpretable descriptions for value type constraints, the method may verify the given value and fail on boundary violations. This may include verification of a link target PID if the property range is another PID record (i.e., a Persistent Entity). If the PID record already contains a value for the given property, the method will overwrite it. Obviously, variations of this method may be provided to replace a complete PID record.

QUERY PROPERTY FROM PID RECORD: Given an object PID and a property PID, the method returns the associated property value or fails if there is no such value defined for this object.

QUERY PID RECORD BY PROFILE: Given an object PID and a profile PID, the method returns all mandatory and optional properties that are available from the given PID record. Essentially, this is a PID record sub-setting operation. The method may fail if the record does not provide values for all mandatory properties. In this regard, the method doubles as a weak conformance check.

WEAK PROFILE CONFORMANCE CHECK: Given a profile PID, the method returns true if all property keys specified in the profile are present in the PID record and if there is a value associated with each of these keys. This method provides the essential facility to type objects.

STRONG PROFILE CONFORMANCE CHECK: Given a profile PID, the method returns true if, in addition to weak conformance, the property values are in the correct range of the respective registered properties. This method is optional since a typing layer implementation will only be able to answer such queries

if the value ranges available from the type registry are described in an interoperable and machine-interpretable manner.

DETERMINE PID CLASS: Since the typing layer introduces PIDs for different kinds of objects, it must also be able to tell a client which kind of object a given PID identifies. Such a method may thus rely on the typing facilities itself to distinguish registered properties, profiles, value types and arbitrary objects from each other.

From an operational perspective, scalability is a potential issue. If there are large numbers of PIDs registered and a lot of queries are issued to the typing layer to determine object types, an implementation should locally cache the type registry contents. This should work well because the type registry contents can assumed to be static over sufficiently long time spans. Additionally, an implementation should be constructed as a distributed system, independent from the requirement to have a single overseeing authority, to offer load-balancing and high availability.

At the organizational level, the entity maintaining the type registry may choose to enforce a central type governance process as described in section 4.3.3. This will then be similar to a continuous and long-lasting ontology engineering exercise; user communities must be involved to mediate conflicts between registered and proposed types, and the goal of preventing too large a diversion between types must be emphasized without sacrificing too much of the specific community use cases. To support these processes, it should offer searching facilities across a full catalog of registered properties, profiles and value types. In principle, the type registry should be open to anyone wishing to register types, since usage of PID systems is often a bottom-up process. Role models for the central overseeing authority are for instance the IANA controlling the IP address namespace and the ITU-T controlling telephone number country codes. The Digital Object Numbering Authority (DONA) which is currently taking over responsibilities for Handle System administration is also a good candidate.

As part of the RDA Working Group initiatives, a type registry prototype that offers some of the required functionality has been proposed by CNRI and is still under development at the time of writing. This prototype offers generic methods to register types and retrieve their definitions, and it appears likely that it can provide at least a value type mechanism as described. However, it is unclear whether it will provide all of the necessary functionality for properties and profiles. The typing API prototype developed during the course of the RDA Working Group on PID Information Types therefore relies on this prototype but also offers additional mechanisms to feed property and profile descriptions into the type registry prototype. For

security reasons, it does not offer public methods to register profiles, properties and value types. The question whether such methods are really required at an automatable web service level very much depends on the scale of adoption by communities.

Some concrete type examples can be found in the next section as well as in table 4 in chapter 6.

5.3 COLLECTION LAYER

As mentioned in section 4.4, collections should ultimately be implemented as common Abstract Data Types. This section briefly discusses exemplary ADT implementations that can conform to all of the collection criteria. In Weigel et al. [97], a stand-alone implementation has been presented that provides List, Set and Map instances based on Handle System records, but does not use the typing facilities presented here. The registered properties introduced in this section are also summarized in table 3.

The particular ADT implementation must be determined at the first stage of the collection process, because some of the ADTs are ordered (lists) and others are unordered (sets). Another factor for the concrete choice of implementation is the desired trade-off on operational complexity. Linked lists offer constant-time insert and extension operations, but linear-time access by index, while arrays offer indexed access in constant time but linear-time inserts. Sets and maps are best implemented as hash maps. All of these implementations share the requirement that the structural contents of instances must be served through PID records, so that if the secondary level of preservation (section 4.2.3) fails, the instances are still fully functional in accordance with a PID system that fulfills criterion 3. This also means that every collection object receives an individual head PID, fulfilling collection criterion 3.

All implementations assume that the persistency layer offers a constant-time method to access PID record entries through their keys. All implementations obey the fundamental requirement that access occurs primarily via single PIDs, which is a core assumption for the whole work (cf. section 1.2).

For **set** and **map** implementations, a hash map is stored in the head PID record, encoded through the key-value pairs. The values will be the member PIDs, while the keys will not be registered properties, but used as integer hash keys for open addressing hash mapping with simple linear probing so that constant-time look-up is ensured. The member PID names act as the input to the hash function. Because PID names may be very similar to each other, particularly if a collection is generated from sequential output, the hash function should try to avoid collisions on very small self-similar input strings. This is not necessarily true for the map implementation, because here, the input

PROPERTY NAME	RANGE	CARDINALITY	PURPOSE
COLLECTION-TYPE	String or CV	1	Indicates for a head PID which type of collection instance is managed through it. Possible values for a controlled vocabulary (CV) include set, map, linked list and array.
MEMBER-OF	Identifier	N	Points from any member to the head PID of a collection.
LINKED-LIST-PREDECESSOR	Identifier	N	Points to the predecessor in a linked list.
LINKED-LIST-SUCCESSOR	Identifier	N	Points to the successor in a linked list.
LIST-HEAD	Identifier	1	Points from a head PID record to the first element in a linked list. May also be useful for array instances to support interoperability.
LIST-TAIL	Identifier	1	Points from a head PID record to the last element in a linked list. May also be useful for array instances to support interoperability.
TOTAL-NUMBER-OF-ELEMENTS	Integer	1	Stores the total number of elements of a collection of any type in its head PID.
READ-ONLY	Boolean	1	If set to true on a head PID, indicates that modification methods should refuse to further change the collection.

Table 3: Summary of all registered properties required to maintain collections. Instead of accessing their values directly, users will call the appropriate collection methods. If collection criterion 1 need not be fulfilled, the cardinality of MEMBER-OF, LINKED-LIST-PREDECESSOR and LINKED-LIST-SUCCESSOR will be 1.

strings will be arbitrary values given by the user. To ensure collection criterion 4, any insert operation must modify the member PID records so it points to the head PID. Ensuring criterion 1 requires that at each element, the type of implementation is recorded as well. This will result in specific registered properties such as `MEMBER-OF` and possibly derived properties to indicate the implementation type.

A **doubly linked list** implementation will store predecessor and successor PIDs in the PID records of the member elements to enable constant-time iterations, using specific registered properties such as `LINKED-LIST-PREDECESSOR` and `LINKED-LIST-SUCCESSOR`. A linked list implementation may work particularly well without requiring collection criterion 3 to be matched. While persistently referencing the full collection will then not be possible, the common operations such as insert, iteration and removal will be possible if at least one element of the linked list is known. If a head PID is assigned, the corresponding PID record should point to the head and tail elements of the linked list to allow flexible iteration from both ends. Also, the record should contain information on the total number of elements. The registered properties for this will be `LIST-HEAD`, `LIST-TAIL` and `TOTAL-NUMBER-OF-ELEMENTS`. If collection criterion 1 should be fulfilled so that a list element can be member of two or more linked lists at the same time but without assigning a head PID, there must be an indication in the record which predecessor and successor links refer to which list instance.

An **array** implementation will use the keys of a head PID record as array indices to store the respective element PIDs. A `MEMBER-OF` value must be stored at element records to fulfill collection criterion 4. The head record may also contain a `TOTAL-NUMBER-OF-ELEMENTS` property.

All implementations described are able to work as dynamic collections providing collection states after initial creation. If a head PID record is available, a `READ-ONLY` property with Boolean value type may be set to mark the fixation stage. All insert, append and remove operations must check the flag and fail if it is active. The head record should also contain a property `COLLECTION-TYPE` to make it easier for services to offer the adequate set of collection methods. Because linked lists and arrays are implementations of the same concept — an ordered collection — it may be possible to migrate from one to the other even after collection creation at the dynamic extension stage. This is however not strictly true (lacking additional knowledge on item ordering) for migrating e.g. a set instance to a linked list or array.

A full exemplary implementation is described in [97]. The implementation is based on the Handle System and makes extensive use of its Handle Index fields to provide constant-time access to PID record values. Because the requirements listed in [97] are equivalent to the

collection criteria defined in section 4.4.3, the exemplary prototype fulfills all of them.

5.4 DISCUSSION

For each of the layers described in chapter 4, possible implementations exist or can be developed. The persistency layer and with it the preservation-centric model are in the responsibility of typical PID systems and their providing entities. Popular existing PID systems exhibit differences in conformance to the fundamental criteria of the persistency layer, though sometimes explicit policies are missing. The widespread ambiguity in meeting criteria 3 and 6 should be seen as an invitation to PID system providers to clearly state their take on the relevant issues such as continued record resolution, identifier re-assignment and change management of identified resources. If the typical uses cases for the individual systems do not demand conformance to a particular criterion and may actually be inhibited by it, such as in the case of acceptable resource replacement, this should be advertised unambiguously, also in view of the functionality provided by typing and collection layer implementations.

A solution fulfilling all criteria can be implemented for example by the DOI System with additional policies that ensure the resolution of PID records. The Handle System is also a good candidate if a set of policies similar to the DOI System is enforced and a proper organizational framework is defined. A possible role model for this is the European Persistent Identifier Consortium (EPIC), which is expected to formulate adequate policies in the future. In conclusion, most of the reasons why criteria may not be met are indeed due to policy issues revolving around the persistency of records and resources, and ensuring these requires a central and well-accepted overseeing authority. An evaluation along the criteria is therefore also always qualitative and informed by weighing different aspects. A technical aspect not described by the fundamental criteria is that any PID system that aims to provide services for huge numbers of PIDs must provide some form of practical scalability both in terms of PID resolution to resources and to PID records. PID records should also exhibit key-based constant look-up times.

A typing layer implementation relies on a system architecture that includes a type registry, an overseeing authority and underlying PID systems. An exemplary prototypical implementation of this architecture is available through the outcomes of the RDA working groups on PID Information Types and Type Registries. The most important non-technical aspect for a sustainable typing layer implementation lies in the governance process of the type registry and the acceptance of its authoritative role.

A collection layer implementation can be implemented through typical actionable abstract data type instances such as lists and sets. As is typical for ADT implementations, no optimal solution for ordered lists can be found as efficiency trade-off decisions between insert and look-up operations must be made. A core benefit of such an approach is however that the concept of ADTs is well-known to any programmer, and chances are that such a solution is well-received by adopters that do not want to get involved in the intricacies of PID systems.

The implementations presented in this chapter also obey the fundamental principles of the layer cake, where underlying layers are functionally independent from higher layers. PID systems have been around for a long time and work well without type registries. The typing architecture (see figure 7) makes use of the persistent identification facilities of the lower layer. The collection layer finally employs a number of types to maintain the structure of the ADT implementations and relies on the constant look-up time of keys in PID records. The rather loose coupling of the layer components through interfaces is quite intentional to allow modular exchange and support an approach where there are distinct service providers for separate layers. The type registry and separate PID systems are a first example for this.

Finally, a sophisticated *collection-enabled PID resolver* would not be situated at the preservation layer, but instead make use of functionality from all three layers. As described in section 4.2, PID systems (as persistency layer implementations) provide two fundamental operations: resolution of PIDs to resources and to PID records. ADT instances that offer head PIDs are identified through them. A collection-enabled PID resolver offers both the low-level operations as well as generic read-only collection operations. Because the necessary properties to hold collection information are registered in the type registry, the resolver will be able to interpret them independent from the original PID generator. For list head PIDs, the resolver can provide access to the first and last element, the list size and access to elements by index. For list elements, it can provide access to previous and next elements. For set head PIDs, it can provide the set size.

6 ANALYSIS

The mechanisms presented in chapter 4 and 5 can be evaluated in terms of how they can enable the use cases presented in chapter 3. In some cases, decisions must be made on important trade-offs and implementation alternatives. A set of overarching tools can finally demonstrate the added value that may be necessary to encourage wide adoption of the conceptual framework presented in chapter 4.

6.1 REFERENCING PRELIMINARY DATA

To provide persistent references for preliminary data that may not be preserved on longer timescales, PIDs can be used if the model of the primary and secondary levels of preservation is observed. Figure 8 generalizes the process described in the use case. When a dataset is ready to be published on ESGF, the publication process scripts should automatically register a PID with a preserving PID system. The initial assignment generates a Persistent Entity that can eventually last longer than the Digital Object that includes the data. The PID does not fulfill the common criteria to be citable (see section 2.4.2) since the data it refers to is deliberately not guaranteed to persist and the identifier may not be widely accepted by publishers and scientists as part of a formally correct reference list. Most importantly, there will not be a standardized set of citation-relevant metadata associated with the identifier, such as author information, dataset title and publisher name (cf. [22]), as these are rather unstable or difficult to obtain at this early phase. A registered property `PUBLICATION-DATE` may however be part of the PID record as it contains static information easy to record automatically as part of the ESGF publishing process.

The PIDs for preliminary data, while not citable, still serve significant practical purposes as indicated in the use case description. If a scientist wishes to refer unambiguously to an ESGF dataset at a more fine-granular level than available through the DOIs assigned, current practice would be to use its name (which is not necessarily unambiguous or consistent across projects publishing on ESGF) or surrogate identifiers such as the "file tracking ID" (also see figure 4), a low-level technical identifier that is unique but not subject to any form of governance and may vanish easily because it can only be resolved through the ESGF metadata catalogs. The automatic assignment of PIDs during ESGF publishing fills this gap.

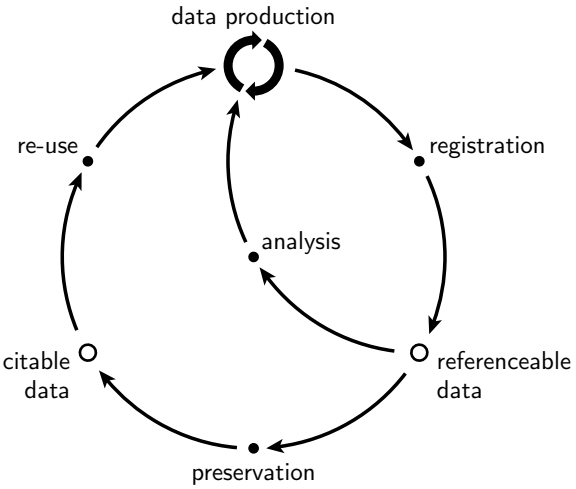

Figure 8: An abstract model of the ESGF data cycle with states and transition actions. Data that leaves the production stage will be registered with a PID so that it becomes referenceable. Further analysis may cause the data to be re-computed, closing the cycle short. Eventually, data passes the final QC checks and is subjected to preservation, usually causing a DOI to be assigned so that it becomes citable and may re-enter the production phase as part of subsequent re-use.

To ensure that the data remain unchanged as long as they are available (criterion 6), certain policies must be enforced within ESGF. The first step for this is to include a `CHECKSUM` property in the PID record to make it possible to detect impermissible changes. As data pass certain gateways throughout the ESGF processes, the checksum must be verified automatically and upon verification failure, data must be rejected. Secondly, the PID should be included in the netCDF header data of the respective files to make detection easier. A user who retrieved a file from the ESGF data space and stored it locally should be able to access the associated metadata records on ESGF using the PID as well. Both aspects are also important for the solution for the versioning use case described further below.

The established process of assigning DataCite DOIs at the eventual preservation phase remains unaffected by the additional identifiers since the unique identification relation is not considered to be functional (cf. section 4.1). To distinguish the early PIDs from the fully citable ones, a globally registered property `CITABLE` should be included in the respective PID records as a marker entry. Additionally, preservation statements may be included as are for example used by the ARK system. Alternatively, the referenceable PIDs may be converted directly into DOIs, keeping the same identifier name, which is a process currently unclear with both technical and organizational challenges remaining unsolved. Technically, the inclusion of single Handles in the continuous DataCite namespace is difficult

because the Handle System uses the prefix/suffix concept to separate namespaces, yet the referenceable PIDs will not share the same prefix. Organizationally, the hand-over of responsibility for the particular PIDs must be clearly defined.

6.2 ACCESS TO SPECIFIC VERSIONS OF A DATASET

The initial problem of the versioning use case can be described with figure 8 when a dataset has gone through the small cycle at least once and enters the registration action again. For versioning to be properly controlled, the publication process must be able to recognize whether a previous version existed. This may be done either by submitting the PID of the old dataset or — if still available — an old file whose header will contain the PID. It might also be done through a separate database, which will entail additional maintenance effort, or through an ESGF catalog search, which may however return ambiguous results and should therefore be avoided.

The process specific to ESGF that takes care of the new publication will do the following steps:

1. Register a PID for the new dataset because the data have changed (cf. criterion 6). Usually also include a `CHECKSUM` property in the record.
2. Optional: Retract the old data from the ESGF data space, but keep the PID. The target resource of the PID will thus become invalid.
3. Put a Boolean flag property `TOMBSTONED` in the old PID record to mark that the resource has been purposefully removed.
4. In the old record, include a property `NEXT-VERSION` with the new PID as value and a property `OBSOLESCENCE-DATE` with the current date.
5. Optional: Put a property `PREVIOUS-VERSION` in the new PID record with the old PID as value. This is not strictly required to access the latest version of a dataset, however it is easily done and may still be useful in view of e.g. provenance tracing tools.

When a user accesses the PID of a dataset that is obsolete, the resolution will fail because the resource is gone. The capabilities of the application-level resolution service must therefore be extended so that it checks the resource before redirecting the user to it and if it recognizes that the resource is gone, it further processes the PID record. Because the properties of the record are globally registered, the service can interpret them correctly even across infrastructures and PID generators. It will recognize the tombstone marker and

inform the user that the dataset has been deleted on purpose and that a new version can be found at the alternative location given by the `NEXT-VERSION` property value. It will also include other useful information such as the obsolescence date. The service may even follow the consecutive chain of PIDs recursively until it encounters a record where no tombstone marker is set and present its resource location to the user.

The use case also describes an additional scenario where it is possible to always retrieve the latest version of a dataset through a separate PID. A solution that covers this case is to extend the implicit notion of a linked list made from the `PREVIOUS-VERSION` and `NEXT-VERSION` entries to a full ADT instance with a separate head PID (collection criterion 3 must be met). This ADT instance must be created when the first version is published, so that the head PID can already be used. The head PID record will not point to an actual target resource, but instead contain a behavioral marker flag `REDIRECT-TO-LAST-ELEMENT`. When resolving the head PID, the resolution service will recognize the ADT instance from the type information in the record, encounter the marker flag and redirect to the last list element accordingly. If collection criterion 4 is not met, it will be impossible to navigate to the head PID from an element PID. In a linked list, this will only cause the look-up operation of the latest version from any arbitrary element to take linear time rather than constant time. If instead an array implementation is chosen, navigating to the latest element will become impossible because array elements are not linked through each others' PID records.

The ESGF workflow described above can be further generalized for cases where there is an arbitrary repository receiving a new dataset that may be either a new version of existing data with the same purpose or a derived data product with a different purpose. This scenario is described for the provenance use case in more detail in section 6.4.

6.3 REFERENCING CUSTOM DATA SLICES

The key component to enable this use case is the bundler service to which the user submits a large number of PIDs for data from various sources. This can be done for instance by providing the list of PIDs directly or submitting a directory containing netCDF files from whose headers the service can glean the PIDs. The bundler service will create a set ADT instance and add all PIDs provided to it. If the collection properties are registered in the type registry and other e-infrastructures offer collection layer implementations as well, the bundle can easily cover input across them. The set ADT instance must bear a head PID (collection criterion 3) so that a single identifier can be included in an article's reference list as opposed to a large number

of DOIs. Because the bundler service works on demand for individual users' requests, the same data product may be included in multiple bundles. Therefore, collection criterion 1 must be met. If it should be possible for a user to bind together bundles already created earlier by other users, the collection layer implementation must conform to collection criterion 2.

If the versioning use case is covered, the bundler can confirm that every PID provided points to the latest version of a dataset by accessing the PID record and determining whether there is a `NEXT-VERSION` property present. Accidental cases where there are two elements that are related by a chain of versions can be easily filtered out as well. It can also verify access to the resource simultaneously and may even retrieve the dataset and verify it against the `CHECKSUM` property value.

Later, when a third party resolves the head PID, a specialized application service queries its PID record, detects the set instance and returns a listing of all member PIDs. It can also access the member PID records and use the mechanisms of the versioning use case to provide information on the resource status and possible follow-up versions that have been published since the bundle was created.

A possible danger for such a solution is of course that any user can request an arbitrary number of PIDs. To prevent misuse and exploitation through malicious bots, adequate security measures must be implemented. A quite sophisticated solution could also hash the identifier names used for creating a particular set and deposit the hash in a database. Later, when a collection for the same set of objects is requested, the existing collection can be reused.

6.4 PROVENANCE TRACING

As described in the use case, derived data objects may be published at repositories different from those hosting the source objects, and this can happen multiple consecutive times. Let us assume that every repository assigns individual PIDs to the derived objects. To record provenance, typed properties should be written in the respective PID records. The generic process is the same as for the versioning use case (see figure 9). The defining difference between versioning and provenance scenarios lies in the intended purpose of the data. A new version of a dataset has the same purpose as the original data; for provenance, the purposes differ. Because the process cannot infer this from the available information, it must be determined a priori whether the data is regarded as a derivation and the provenance scenario applies. This may be easier if the process is performed by data derivation tools rather than by the receiving repositories because by using such a tool, it is clear that the result will be a product with a new purpose.

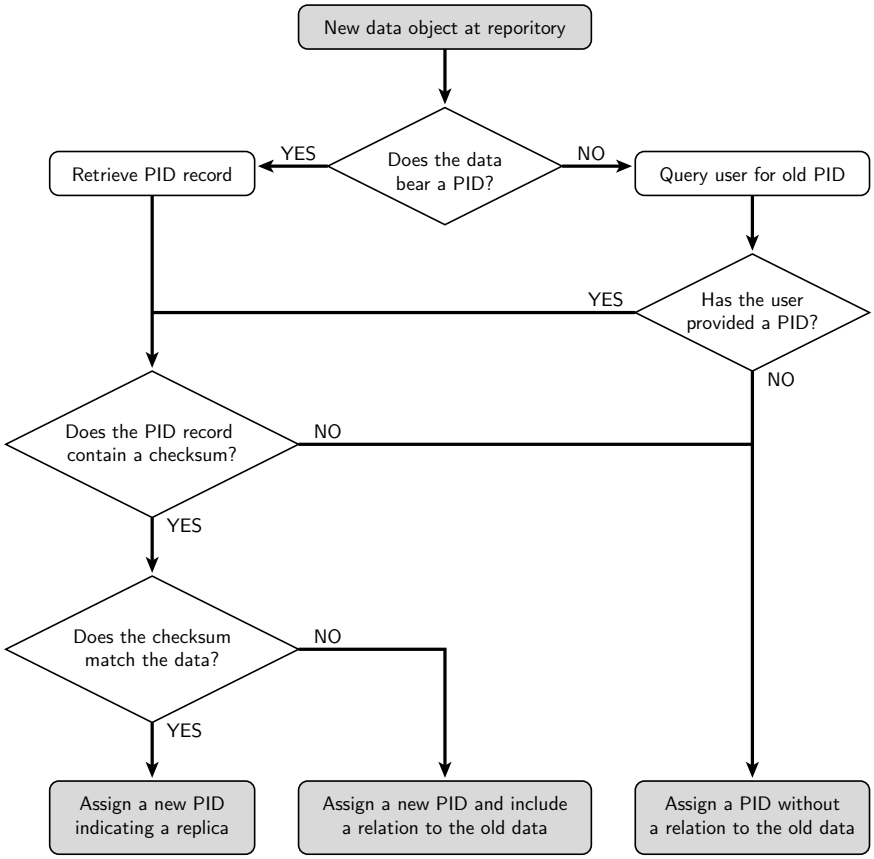

Figure 9: When a new data object arrives at a repository, different outcomes are possible regarding the assignment of a new PID for it. Because the generic process is the same for versioning and provenance scenarios, the particular meaning must be determined a priori.

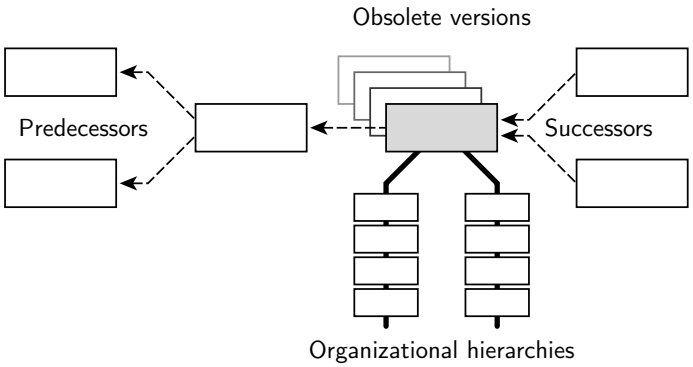

Figure 10: In combination, the use cases for versioning, provenance and custom slices generate a PID graph with three main axes. The relations may be encoded as typed links either stand-alone or as part of collection instances. Successors may not be available in all cases.

Figure 9 describes the process at the repository or tool in more detail. Data may bear a PID e.g. in the form of a netCDF header field value. The provenance relation stored in the new PID recorded is a property of type `PREDECESSOR`. If the user supplies multiple former PIDs, all of them should be included as predecessors.

The graph resulting from these predecessor relations will be a proper directed acyclic provenance graph as described by Moreau [65]. All its edges will point to predecessor objects, and thus, it will be possible to navigate the graph structure to all ancestors, but not to any descendant. To achieve this, the publication process must generate `SUCCESSOR` property entries in the original objects' PID records as well. This may cause issues if some objects are re-used so often that the number of successor entries may become too large to manage adequately. In any case, a generic service tool can be implemented that relies solely on the registered properties to trace the provenance of data across disciplinary and infrastructural boundaries. Additional provenance records, e.g. from workflow engine logs, may be pulled in if they are referenced in the PID records as well through a distinct property. If the two levels of preservation are observed, resources and even whole repositories may become inaccessible without breaking the provenance graph.

6.5 DISCUSSION

If all use cases are combined, the resulting structure embedded in the PID records forms a proper PID graph. Figure 10 explains how the different use cases generate edges between data objects along three thematic axes for versioning, provenance and custom slices. The use case dealing with custom slices may be further abstracted to generally reflect the multiple organizational hierarchies in which data

may be organized as already indicated by the use case description in section 3.4. The individual solutions for the use cases already contain some essential added-value services, such as a service that redirects to the latest version of a dataset or a generic provenance tracing service. One can easily imagine that either more of such specialized services could be created for other use cases or that a common "PID information service" can be created that interprets the PID records based on the most commonly used and registered properties. Such a service should allow a user to navigate to later and possibly former versions, move along the provenance graph, and to dig down into possible sub-elements. Where individual resources are not available anymore, the service should offer tombstone pages with the remaining information.

A collection layer implementation that conforms to all collection criteria is not required for all use cases. Of course, an ideal solution would provide functionality that matches all criteria. However, such a solution is the most expensive one to build, and therefore, a simpler solution may work well if only parts of the use cases must be met. The detailed descriptions provided in this chapter should help to decide on such trade-offs. For example, the versioning use case can work very well with a solution that does not support multiple membership of elements (collection criterion 1).

Table 4 summarizes the registered properties required to enable all use cases. Cardinalities may be stored in the registration records along with the property names and can be used by verification tools to warn about inconsistent PID records. Alternatively, it is possible to include the cardinalities as part of specific profiles for the individual usage scenarios. This may be beneficial for example to describe the versioning of source code states where there is more than one prior version due to a merge action of two converging branches. Source code may be persistently identified to exactly define the model used to compute a particular data object.

The question of what other information to store in a PID record and access via registered properties always depends on the specific disciplinary use cases and therefore poses a wider area for future cross-disciplinary activities. A good rule of thumb to decide on the inclusion of a particular property may be to estimate whether its value changes over time, because modifying existing PID records is best avoided, but also to focus on cross-disciplinary usage and machine agents. In addition to the detailed properties listed in the table, further suggestions include the owner of an object (a string or even an identifier, e.g. provided through ORCID), the license terms for the object and the data format or encoding (both preferably as controlled vocabulary terms).

Typically, provenance information is stored in the form of more or less formal documents, forming a spectrum from simple log files to

standardized formalizations such as the W3C PROV standard [66]. Such solutions are often insular, and there is no coherent workflow trace. Missier et al. [62] describe an abstract model for provenance traces that can provide a continuous trace across systems if the connections between the different input and output objects can be established. Their solution is to use a common provenance model and map each individual trace to it. As they correctly remark, identifying the common elements of the individual system traces is however challenging if the same objects in different traces have different names. Their solution to this challenge is to record the possible changes that take place unobserved by provenance tracing, such as copying data from a local to a global store, must be recorded as additional relations in a provenance trace of the data sharing process. When PIDs are assigned to objects and interlinked by relations in the PID records, these relations will be available so that a service capable of mapping individual traces can produce a continuous trace as described by Missier et al.

PROPERTY NAME	RANGE	CARDINALITY	PURPOSE
CHECKSUM	String	1	Stores a checksum calculated over the referenced data, enabling authenticity checks. The type of checksum should be indicated in the registered property's record. Multiple checksums (e.g. MD5, SHA-1, SHA-2) may exist for a resource as separate properties with distinct names.
CITABLE	Boolean	1	Indicates whether the PID should be cited in formal reference lists, subject to diverse policies.
NEXT-VERSION	Identifier	1	Points to the next version of a data object.
PREVIOUS-VERSION	Identifier	1	Points to the previous version of a data object.
REDIRECT-TO-LAST-ELEMENT	Boolean	1	If set, indicates that upon resolution of a list head PID, the user should be redirected to its last element.
PUBLICATION-DATE	Date	1	Indicates when the PID was assigned.
OBSOLESCENCE-DATE	Date	1	Indicates when the referenced object has been obsoleted by a new version.
TOMBSTONED	Boolean	1	If the referenced data object is missing and the value is true, indicates that it was deleted on purpose. If false, this may be a sign for a broken resource or archive.
PREDECESSOR	Identifier	N	Points to one of the data object's predecessors.
SUCCESSOR	Identifier	N	Points to derived data.

Table 4: Possible registered property types for implementing the use cases as described in this chapter.

7 RELATED WORK

Exemplary PID systems have been discussed in section 5.1. This chapter focuses on applications of these and other systems in a broader context.

7.1 LINKED DATA

The Resource Description Framework (RDF) [78] is a series of W3C specifications that describe in its core a graph-based data model for representing information about web resources. RDF documents consist of large number of *triples*, each of which is a statement consisting of a subject, a predicate and an object, which are usually URIs. Objects may also be *literals* to convey values of a specified data type. The essential data types include many of the standard XML Schema data types, but custom data types may also be defined. RDF is principally independent from a particular encoding, however such a serialization is required so that RDF data can be read and exchanged. The W3C has standardized two encodings, RDF/XML and RDFa (RDF triples embedded in HTML documents).

The goal of Linked Data is, roughly, to create typed links between data from different sources on the web [14, p. 2]. Such data should be machine-readable, linked to data from third parties and linkable by them and have its semantics explicitly defined. This is achieved by using RDF triples whose constituents come from different namespaces and where the predicates form the typed links. The idea of Linked Data is usually summarized in the four so-called *Linked Data principles* [14, p.2][1], which are stated literally as:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things

Adhering to the principles should allow anyone to publish rich machine-interpretable information using the World Wide Web as

1 Also originally formulated by Berners-Lee (2006): Linked Data – Design Issues, `http://www.w3.org/DesignIssues/LinkedData.html`, last checked Feb. 27, 2015

the transport medium and conforming to its fundamental architecture and standards. The fundamental model builds on top of the decentralized nature of the World Wide Web, where registration and governance of RDF documents and the URIs used in them is intentionally not required. The early adopters of Linked Data whose collective data forms a single continuous RDF graph have been summarized under the popular notion of the Linked Open Data (LOD) cloud.

So although there is no central coordination and governance, Guéret et al. [35] detect a small number of nodes whose failure may very likely break the overall data graph. For the 2010 study and state of the Linked Data graph, the top three host name nodes were `xmlns.com`, `dbpedia.org` and PURL.org. `xmlns.com` is the home namespace of the "Friend of a Friend" (FOAF) RDF vocabulary [34][2] which can be used to encode information about individuals and their relationships and DBpedia is a large RDF dataset of information from Wikipedia across multiple languages [5].

For scientific publications, RDF/XML encoded metadata is provided by Crossref.org via content negotiation mechanisms when resolving DOIs through `http://data.crossref.org`. DataCite offers a similar service at `http://data.datacite.org` for at least parts of its metadata. While Linked Data does not follow the preservation policies motivating the use of persistent identifiers, RDF-encoded relations between resources are a flexible enough concept that can also work on top of the information contained in the collection and typing layers.

The notion of a PID graph discussed in section 4.5 can ultimately bridge the gap between the basic information available from Persistent Entities and the rich and possibly machine-interpretable information available through Linked Data. The PID graph can be encoded through RDF properties and with a proper HTTP-URI based interface implementation, PIDs can act as RDF subjects and objects. Making the information contained in the PID graph available to use by third parties can potentially enable a vast range of usage scenarios found in literature on Linked Data, including provenance tracing and recombination of data. Due to the underlying foundational layer, information present in this "PID cloud" should be suitable to address the sustainability concerns raised by Guéret et al. While by far not all data available from the LOD cloud today may be served as part of context information from typing and collection layer implementations, the PID cloud information can act as a reliable fall-back for the most essential binding elements. A necessary precondition for establishing such a solution is however a unification of the relation information available from type registries and commonly used ontologies.

2 The latest vocabulary specification may be found at `http://xmlns.com/foaf/spec`, last checked Feb. 27, 2015

7.2 THE ENTITY NAME SYSTEM

The Entity Name System (ENS) is described in a PhD thesis [86] and a series of articles [15, 16, 6] as a comprehensive identification solution that aims to combine principles and advantages from both Linked Data and PID systems. As Bazzanella et al. [6] remark, there are important differences between Linked Data URIs and Persistent Identifiers, and, as the authors state, one may even detect a "subtle underlying hostility" between the two approaches and schools of thought that have developed in parallel since the mid of the 1990s. The Entity Name System aims to provide interoperability not only across PID providers, thus addressing a concern of this thesis' original questions, but also between the PID world and the Linked Data world. Bazzanella et al. emphasize that the issue of persistent identification can only be adequately approached when including issues of policies and responsibilities that underlie the maintenance of PID systems, a factor which is claimed to have been ignored in the Semantic Web community so far [15, 86].

Bazzanella et al. compare PID and Linked Data solutions in large detail. The article provides a good overview of the different viewpoints from the possibly conflicting schools of thought. The most important differences in view of the discussion provided in this thesis can be summarized as follows:

NAMING AUTHORITIES AND TRUST: PIDs inherently rely on naming authorities that establish the necessary trust framework for long-term PID resolution, while Linked Data URIs thrive on the decentralized and ungoverned nature of the World Wide Web: Anyone can publish Linked Data URIs. Bazzanella et al. however fail to note that to provide URIs in the first place, one has to acquire (register, buy) a name within the Domain Name System. This act is concededly simpler and comes with less contractual obligations than acquiring a PID from e.g. the DOI system, but it still cannot be ignored in terms of resources to spend.

CROSS LINKAGE: A defining goal of Linked Data is to establish machine-interpretable formal relations between resources, which Bazzanella et al. describe as missing in the contemporary PID systems.

BUSINESS MODELS: Some PID systems (most notably, the DOI System) require some costs to be covered before PIDs can be assigned, while Bazzanella et al. state that Linked Data URIs come virtually free of charge. This must however be examined a bit more critical than in the article by Bazzanella et al. Usually, resolving HTTP URIs requires a web server to be maintained and a domain name to be acquired as already mentioned above.

These costs may not be visible to the individual developer, but cannot be ignored on the longer term. As explained in section 2.4.2, URIs do not qualify as persistent identifiers because the responsibilities for long-term maintenance are unclear. Neglecting these responsibilities certainly reduces the costs involved, but does not adequately address the question of total costs.

PERSISTENCY: Bazzanella et al. discuss the possible stability of Linked Data URIs in detail and refer to a notion originally described by Tim Berners-Lee in 1998 that URIs should be stable by designing them adequately[3]. Although this, in principle, shows some intention shared with persistent identifiers, Bazzanella et al. continue to describe the drawbacks to this approach from the standpoint of the traditional PID users. Besides the issue of lacking a trusted naming authority, this includes the problem that web resources may change uncontrolled if they are dynamic (cf. criterion 6).

A central design constraint for the ENS has been the motivation to avoid multiple identifiers for the same object (cf. [15, p. 554]), an interpretation of strict uniqueness (see section 4.1). This is done by checking the essential characteristics of a resource prior to assigning a new identifier for it. The ENS requests and stores the characteristics in the form of metadata that are analyzed whenever an identifier is requested and if the characteristics match, the existing identifier will be returned. Central to this is the notion of entity types such as "person", "organization" and "event" [15, p. 557]. These resemble a form of typing layer profiles which have to be agreed upon first or, alternatively, are brought in by referencing external schemata. The actual information characterizing an entity is gathered upon first assignment of an identifier and may be extended later on. As Bouquet et al. [15] remark, this may invalidate the original decisions about the identity of two requested objects in hindsight, which in consequence causes new equivalence relations to be established between objects formerly considered separate or divides formerly identical objects. Although Bouquet et al. describe that such a process may be difficult due to external use of the identifiers in the wild, they unfortunately do not explain further how such an approach matches the original intention to avoid multiple identifiers for the same entity. Given the available literature, it remains unclear whether this is actually practically achievable with the proposed system since there are no long-term quantitative studies available. An alternative but weaker description of the central service offered by the ENS is therefore that it provides assertions (which may change over time) on the possible equivalence of resources given a set of identifiers.

3 The original source may be found at `http://www.w3.org/Provider/Style/URI`, last checked Feb. 27, 2015

From the literature it remains unclear exactly which characteristics are essential for stating that two resources are identical. The algorithm for matching entities is based on syntactic rather than semantic analysis [16, p. 263]. The notion of an entity description consisting of a set of key-value pairs [86, p. 33] is similar to the notion of PID records and it may be made accessible from the outside given an identifier known to the system.

As Bazzanella et al. state, the ENS does not aim to be another fully-fledged identifier system, but should instead provide interoperability between existing approaches. Therefore, it is unsurprising that some of the criteria of the foundational layer are not met, most importantly 1 since ENS identifiers are also used as the resource locators. The ENS thus in fact does not qualify as a PID system as defined throughout chapter 4 — it does not facilitate redirection. Criterion 2 is met since the ENS uses full URIs as identifier names; in consequence, criteria 4 and 5 are met due to the Domain Name System as the central resolver (still not providing redirection). Assuming that access to the entity descriptions can be provided as a separate service, criterion 3 may be met, a definite answer remains however open as the literature lacks more specific details. Criterion 6 is not met because the ENS does not impose restrictions on keeping identified resources intact.

But although the ENS is not designed to be a full PID system, it is a possible candidate for a typing layer implementation. Its notion of fundamental entity types characterized by the entity descriptions provides the essential typing service, however, it remains incomplete with respect to an ideal typing layer since the notions of registered fine-granular properties and a canonical type registry are missing. Regarding the collection layer, the ENS does not offer any notion of actionable collection ADT instances.

7.3 APPROACHES FOR PROVIDING ACTIONABLE COLLECTIONS

At a first glance, some of the navigational capabilities of collection ADT implementations can be offered through structured metadata. The DataCite metadata kernel 3.1 [22], for example, offers a set of elements that establish relations between a DataCite resource and any other entity that bears one of the commonly used identifiers. Starr and Gastl [84] provide a comprehensive discussion of the possible relation types, which includes specifically relations for subsets and versions that are also present in the kernel 3.1. The use cases of versioning and providing custom data slices may be covered by these relations for final datasets that receive such DOIs. However, although a head identifier will be effectively available for the subset case, there is no notion of dedicated ADT operations. A DataCite-specific collection layer implementation may reuse the existing relations, but also requires dedicated services that provide such operations.

The integrated rule-oriented data system (iRODS) [76] aims to facilitate management of distributed scientific data, covering multiple phases across the data life cycle for sharing, publishing and also preserving data. The notion of collections is a central element in iRODS, as all data objects managed by it are arranged in collections which can then be shared. Elements and collections are individually identified. Context information such as provenance records and information on the preservation policies are provided along with the collections. Although a distinct set of ADT implementations with their characterizing complexity trade-offs is not offered by iRODS, iRODS management operations may be directly mapped to common ADT operations if an underlying implementation can offer these. In terms of interoperability, translating the iRODS mechanisms to a collection layer interface may enable the interchangeable use of different PID systems as the basis for iRODS data collections.

7.4 RESEARCH OBJECTS

Bechhofer et al. [8] argue that Linked Data does not adequately meet the requirements of scientists producing and sharing data, because publishing scientific datasets as Linked Data does not provide enough context to ensure scientific reproducibility. As a solution, Bechhofer et al. present the notion of Research Objects (ROs), which are semantically rich compound artifacts that bind together data and context information, most importantly provenance. While all content of a Research Object (data and context) is published as Linked Data, the Research Object metaphor aims to ensure that the overall structure is preserved even if the constituents are distributed across disparate locations and refer to each other through their respective URLs. Bechhofer et al. also describe added-value services that act on whole Research Objects instead of potentially incomplete Linked Data resources. Among the use cases for ROs, Bechhofer et al. also include versioning and basic management routines for ROs (create, retrieve, update, delete) [8, p. 12].

Bechhofer et al. name exemplary parts of a RO, which besides the actual result data includes information on original research questions, methodology, organizational or funding information, input data and result evaluation reports. ROs are also dynamic, as they proceed through the scientific data life cycle, and thus exhibit distinct stages of work in progress, and published or archived work. Bechhofer et al. describe these in terms of three stereotypes: “Live Objects”, “Publication Objects” and “Archived Objects” [8, p. 17]. Archive Objects need not be citable, however both Publication and Archive Objects are described as immutable, and thus other defining differences remain unclear. Bechhofer et al. emphasize that ROs are versioned, however they allow for different models on how this may be achieved,

most notably by either modifying the RO contents while retaining its identity ("in-place update") or by creating a new RO with new identity [8, p. 18]. A third option is implied through the life cycle process from live to archived or published objects.

Bechhofer et al. emphasize that ROs do not aggregate content through duplication but by referencing elements from different sources, following the same ideas that underlie a conformance to collection criterion 1. A RO should also be uniquely identified, similarly to a head PID (see collection criterion 3). Bechhofer et al. describe an implementation of ROs based on Linked Data and "resource maps" from the Open Archive Initiative Object Reuse and Exchange protocol (OAI-ORE)[4].

The facilities offered by the collection and typing layers can provide a suitable framework to construct Research Objects as well. The defining difference between the in-place versioning strategy and versioning by creation of new instances can be described with the conformance of the underlying PID system to criterion 6. If the criterion is enforced, the successively created objects should be arranged in a linked list instance or at least be related to each other with registered types. The RO compound must bear a head PID (collection criterion 3 must be met) and can be implemented for instance as a set ADT instance that aggregates multiple other objects, each with a distinct type given through a registered profile to distinguish e.g. input data, result data and methodology or provenance information. A more sophisticated solution may use a map instance with the registered profiles as keys. The constituent objects can then be either opaque objects retrievable through a PID or further collection instances if nesting is possible (collection criterion 2). The whole complex compound object can then be published by an application layer on top of the collection layer that exposes the typed links and inherent collection relations as RDF-encoded Linked Data relations between PID-based URIs.

7.5 DISCUSSION

The existing approaches presented in this chapter typically cover parts of the overall layered functionality presented in chapter 4, but are not as concise and do not take the particular requirements of persistent identification into full account. Nonetheless, all presented approaches could benefit in some way from a coherent underlying PID framework, or vice versa, the uptake of persistent identifiers and in particular associated essential metadata can benefit from the more sophisticated functionality that particularly Linked Data and Research Objects provide.

4 `www.openarchives.org/ore/1.0/datamodel`, last checked Feb. 27, 2015

8

CONCLUSIONS

This thesis introduces a formal model and a set of conceptual and practical solutions that make it possible to understand persistent identification of digital objects not only as a means to name and retrieve an object over time but also to put it into a persistent context independent from its current storage location. Today, there are numerous activities in computer science dealing with increased virtualization, and further abstraction from storage is one of the possible aspects matching the concepts presented here. An essential part of the abstraction is achieved by relating a persistent identifier with other objects and making the relations interpretable by machine agents. A central design criterion that also sets the presented concepts apart from conventional approaches is that data objects are regarded as being not necessarily persistent, but their identifiers must be kept even beyond their lifetime. In consequence, preservation can be subdivided into a primary and a secondary level, where context information resides at the former and the identified object at the latter. Preserving the identifier however is not an adequate metaphor for preserving the full context. As a solution, the notion of a Persistent Entity has been defined which can be easily preserved even if the identified object is gone. The Persistent Entity provides all means to be dereferenceable through an identifier, keep essential metadata and the relations to other objects.

A second important design factor also mentioned in the EU report from 2010 [79] has been the expectation that persistent identifiers will be created for an increasingly large number of objects. Aside from the facilities to relate objects to each other, the notion of actionable collections of identified objects provides a mechanism to deal with large numbers of objects in automated workflows. Actionable collections are based on the idea of commonly known abstract data types such as lists and sets with the intention to make adoption easier for those familiar with higher-level programming languages and to provide an implementation-agnostic abstract interface. This is particularly important for use cases spanning communities, e-infrastructures and even PID systems. Particularly with respect to interoperability across PID systems, the notion of collections and the role of a type registry and its registered properties are crucial.

The motivating use cases have been taken from the domain of Earth system model data management, where the Earth System Grid Federation (ESGF) is the state-of-the-art e-infrastructure for global distribution of model and increasingly observational data

products. The abstract concepts for PIDs, typing and collections can be exploited to implement the use cases and illustrate a number of tools that provide added value to users, such as provenance and version tracing or a custom "shopping basket model" approach for referencing heterogeneous data collections. Such tools may turn out to be elemental to encourage adoption of persistent identifiers within ESGF and beyond if users demand that such services be provided at larger scale and an adequate level of quality.

8.1 FUTURE WORK

Throughout the chapters, some areas for possible future conceptual and practical development have been already been identified and are summarized in the following.

The use cases presented in chapter 3 have motivated the conceptual design of the framework, yet the concept of persistent identification is not at all bound to the Earth sciences. The framework may be transferred to other domains and even form the basis for a discipline-agnostic solution as also indicated in the problem statement. Whether this can work in practice can only be determined after evaluation by other disciplines.

The formal model defined in section 4.1 is adequate to answer the original research questions and address the selected use cases. Yet a more sophisticated model describing the evolution of identifiers, PID records and referenced objects explicitly over time may be required to address a wider range of use cases or preservation workflows. A possible approach may be to enhance the definitions of ϕ and σ so that they describe sequences of discrete states. The concepts underlying the architectural layers must however be respected, such as the fundamental criteria and the coherence of collections over time. The implications of an enhanced model may finally have impact on operational policies to be enforced by stakeholders such as PID providers and long-term archives in order to uphold the validity of the formal constraints. A related issue is the question of how the equivalency of digital objects is defined, and how to deal with the differences in understanding it from a discipline-specific or a preservation perspective (i.e. preserving the bit-stream or the content across format changes). The exemplary ESGF use cases assume bit-identity, motivated for example by the consequent use of checksums for replication and version checks, but this cannot be assumed for every other application scenario and scientific discipline. It remains open whether and how far a more sophisticated formal model for digital objects can address these issues.

As discussed in section 7.1, offering information associated with persistent identifiers as Linked Data can offer a wide range of third-party applications, particularly with regard to the PID graph that

contains relations between Persistent Entities and collections. For this to work in practice, a proper foundation with formal ontologies underlying the exposed information is required. Since the relations are implemented through properties from the type registry, the type registry contents must be exposed as a proper formal ontology. This is however not a simple technical interface, since the quite sophisticated knowledge required to build a coherent ontology may only be implicit in the type registry contents. It therefore remains unclear how sophisticated such an ontology can be and, in fact, there may be several emerging interconnected ontologies if different communities implement their own type registry instances. Practical adoption must carefully coordinate these processes so that the information available in a Linked Data encoding remains meaningful with respect to the combination with other Linked Data datasets. From the perspective of Linked Data, including persistent identifiers with additional context information in the overall information graph is a valuable achievement as this addresses concerns of information stability in the LOD cloud where loss of a few essential nodes may have vast impact.

Related to this, the initial registered properties summarized in table 3 and table 4 form the nucleus around which other properties and aggregating profiles should be defined with interdisciplinary applications in mind. Section 4.3.3 discussed possible models for implementing a type governance process, yet the ultimate operational structure of a type registry federation remains open. Actionable collections of persistently identified objects may eventually be aligned to mechanisms in other existing systems (e.g. iRODS).

With possibly millions of persistent identifiers being envisioned for applications in EUDAT and the CMIP6 data to be served by ESGF, practical concerns regarding identifier resolution performance and access to PID records must be addressed. It remains unclear how far existing PID systems such as the Handle System will be able to scale to these numbers, particularly if a large number of PIDs must be created in the very short time frame of a few seconds or if a massive number of objects is relocated and location information must be updated for possibly millions of PIDs. A first alternative approach for this may be to use key-value stores that offer eventual consistency such as Dynamo [23] or Apache Cassandra and that are optimized for largely static information that is frequently extended such as in BigTable [19]. The essential conceptual distinction of persistent identifiers that such an approach should exploit are the largely static nature of PID records and the identifier-based singular mode of access. Typically, information is not queried across identifier records; the design of actionable collections presented in section 4.4 respects this view as well. A pragmatic solution should also be lightweight enough so that it can be seamlessly integrated with existing e-science

services, which are often based on REST-style APIs. In this regard, a transparent REST interface that provides highly efficient access to distributed key-value stores may be a promising approach [31].

Another option for a fully distributed PID system that is able to address the scalability challenges may be to use peer-to-peer networking technologies. Bolikowski et al. [58] present an exemplary approach with additional trust mechanisms that aim to prevent forgery. Important questions for implementing such a system include how to ensure its longevity, possibly by building up a critical mass of nodes, how to integrate existing PID systems and how to come to a viable business model that finds the balance between openness of identifier assignment and covering, for example, the most fundamental costs of maintaining the software stack. In terms of the conceptual framework, it would most likely be possible to provide a persistency layer implementation at individual nodes and accordingly provide interoperability with other PID systems.

In general, practical adoption of the presented framework should be made across existing PID systems. Eventually, this may also include identifier systems that do not focus solely on digital data or articles, such as ORCID or IGSN. In this respect, people or geophysical samples can form new elemental types, profiles to be registered with their own sets of properties, and be arranged in interoperable collections with far different entities. The collection layer does not inhibit such usage and a type registry federation should enable such broad usage scenarios as well. In the end, the notion of persistent identification is neither bound to digital objects nor the Earth sciences.

BIBLIOGRAPHY

[1] Panagiotis Adamidis, Irina Fast, and Thomas Ludwig. "Performance Characteristics of Global High-Resolution Ocean (MPIOM) and Atmosphere (ECHAM6) Models on Large-Scale Multicore Cluster." In: *Parallel Computing Technologies*. Ed. by Victor Malyshkin. Vol. 6873. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 390–403. DOI: 10.1007/978-3-642-23178-0_34.

[2] Arthur Allison, James Currall, Michael Moss, and Susan Stuart. "Digital identity matters." In: *Journal of the American Society for Information Science and Technology* 56.4 (Feb. 2005), pp. 364–372. ISSN: 1532-2882. DOI: 10.1002/asi.20112.

[3] Reinhard Altenhöner, Harry Enke, Bernadette Fritzsch, Jens Klump, Michael Lautenschlager, Jens Ludwig, and Heike Neuroth. *Digitale Forschungsdaten bewahren und nutzen – für die Wissenschaft und für die Zukunft*. Network of Expertise in Long-Term Storage of Digital Resources (nestor), 2009.

[4] William Y. Arms. "Uniform resource names: handles, PURLs, and digital object identifiers." In: *Communications of the ACM* 44.5 (May 2001), p. 68. ISSN: 0001-0782. DOI: 10.1145/374308.375358.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. "DBpedia: A Nucleus for a Web of Open Data." In: *The Semantic Web*. Ed. by Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux. Vol. 4825. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. Chap. 52, pp. 722–735. ISBN: 978-3-540-76297-3. DOI: 10.1007/978-3-540-76298-0_52.

[6] Barbara Bazzanella, Stefano Bortoli, and Paolo Bouquet. "Can Persistent Identifiers Be Cool?" In: *International Journal of Digital Curation* 8.1 (June 2013), pp. 14–28. ISSN: 1746-8256. DOI: 10.2218/ijdc.v8i1.246.

[7] Neil Beagrie. "Digital Curation for Science, Digital Libraries, and Individuals." In: *International Journal of Digital Curation* 1.1 (Dec. 2008), pp. 3–16. ISSN: 1746-8256. DOI: 10.2218/ijdc.v1i1.2.

[8] S. Bechhofer, J. Ainsworth, J. Bhagat, I. Buchan, P. Couch, D. Cruickshank, D. D. Roure, M. Delderfield, I. Dunlop, M. Gamble, C. Goble, D. Michaelides, P. Missier, S. Owen, D. Newman, and S. Sufi. "Why Linked Data is Not Enough for Scientists." In: *Sixth IEEE International Conference on e-Science*. Brisbane, Australia: IEEE, Dec. 2010, pp. 300–307. ISBN: 978-1-4244-8957-2. DOI: 10.1109/escience.2010.21.

[9] Fran Berman, Geoffrey C. Fox, and Anthony J. G. Hey, eds. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003. ISBN: 978-0-47085-319-1.

[10] T. Berners-Lee. *RFC 1630: Universal Resource Identifiers in WWW*. IETF, 1994.

[11] T. Berners-Lee, R. Fielding, and L. Masinter. *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*. IETF, 1998.

[12] T. Berners-Lee, R. Fielding, and L. Masinter. *RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*. IETF, 2005.

[13] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, and D. Williams. "The Earth System Grid: Supporting the Next Generation of Climate Modeling Research." In: *Proceedings of the IEEE* 93.3 (2005), pp. 485–495. DOI: 10.1109/jproc.2004.842745.

[14] C. Bizer, T. Heath, and T. Berners-Lee. "Linked Data – The Story So Far." In: *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems* 5.3 (2009). Ed. by T. Heath, M. Hepp, and C. Bizer, pp. 1–22. ISSN: 1552-6283.

[15] P. Bouquet, H. Stoermer, C. Niederee, and A. Maa. "Entity Name System: The Back-Bone of an Open and Scalable Web of Data." In: *Semantic Computing, 2008 IEEE International Conference on*. IEEE, Aug. 2008, pp. 554–561. ISBN: 978-0-7695-3279-0. DOI: 10.1109/icsc.2008.37.

[16] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella. "An Entity Name System (ENS) for the Semantic Web." In: *The Semantic Web: Research and Applications*. Ed. by Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis. Vol. 5021. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. Chap. 21, pp. 258–272. ISBN: 978-3-540-68233-2. DOI: 10.1007/978-3-540-68234-9_21.

[17] Jan Brase, Michael Lautenschlager, and Irina Sens. "The Tenth Anniversary of Assigning DOI Names to Scientific Data and a Five Year History of DataCite." In: *D-Lib Magazine* 21.1/2 (Jan. 2015). ISSN: 1082-9873. DOI: 10.1045/january2015-brase.

[18] Reinhard Budich and Wolfgang Hiller. "ESM Data Archives in Times of the Grid." In: *Earth System Modelling – Volume 6*. SpringerBriefs in Earth System Sciences. Springer Berlin Heidelberg, 2013, pp. 1–3. DOI: 10.1007/978-3-642-37244-5_1.

[19] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. "Bigtable: A Distributed Storage System for Structured Data." In: *ACM Transactions on Computer Systems* 26.2 (June 2008), pp. 1–26. ISSN: 0734-2071. DOI: 10.1145/1365815.1365816.

[20] Luca Cinquini, Daniel Crichton, Chris Mattmann, John Harney, Galen Shipman, Feiyi Wang, Rachana Ananthakrishnan, Neill Miller, Sebastian Denvil, Mark Morgan, Zed Pobre, Gavin M. Bell, Charles Doutriaux, Robert Drach, Dean Williams, Philip Kershaw, Stephen Pascoe, Estanislao Gonzalez, Sandro Fiore, and Roland Schweitzer. "The Earth System Grid Federation: An open infrastructure for access to distributed geospatial data." In: *Future Generation Computer Systems* 36 (July 2014), pp. 400–417. ISSN: 0167739X. DOI: 10.1016/j.future.2013.07.002.

[21] Karen Coyle. "Identifiers: Unique, Persistent, Global." In: *The Journal of Academic Librarianship* 32.4 (July 2006), pp. 428–431. ISSN: 00991333. DOI: 10.1016/j.acalib.2006.04.004.

[22] *DataCite Metadata Schema for the Publication and Citation of Research Data, Version 3.1*. Technical report. DataCite, Oct. 2014. DOI: 10.5438/0010.

[23] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. "Dynamo: Amazon's Highly Available Key-value Store." In: *ACM SIGOPS Operating Systems Review* 41.6 (Oct. 2007), pp. 205–220. ISSN: 0163-5980. DOI: 10.1145/1323293.1294281.

[24] Deutsche Forschungsgemeinschaft, ed. *Sicherung guter wissenschaftlicher Praxis: Safeguarding Good Scientific Practice*. Weinheim: Wiley-VCH, 1998. ISBN: 9783527606252. DOI: 10.1002/3527606254.

[25] *DOI Handbook, version 5*. International DOI Foundation, Aug. 2014. DOI: doi:10.1000/182.

[26] Ruth E. Duerr, Robert R. Downs, Curt Tilmes, Bruce Barkstrom, W. Christopher Lenhardt, Joseph Glassy, Luis E. Bermudez, and Peter Slaughter. "On the utility of identification schemes for digital earth science data: an assessment and recommendations." In: *Earth Science Informatics* 4.3 (Sept. 2011),

pp. 139–160. ISSN: 1865-0473. DOI: 10.1007/s12145-011-0083-6.

[27] Harry Enke, Norman Fiedler, Thomas Fischer, Timo Gnadt, Erik Ketzan, Jens Ludwig, Torsten Rathmann, Gabriel Stöckle, and Florian Schintke. *Leitfaden zum Forschungsdaten-Management: Handreichungen aus dem WissGrid-Projekt*. Ed. by Harry Enke and Jens Ludwig. Glückstadt: Verlag Werner Hülsbusch, 2013. ISBN: 978-3-86488-032-2.

[28] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1*. IETF, 1999.

[29] Roy T. Fielding. "Architectural Styles and the Design of Network-based Software Architectures." PhD thesis. University of California, Irvine, 2000.

[30] Ian Foster, Carl Kesselman, and Steven Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." In: *International Journal of High Performance Computing Applications* 15.3 (Aug. 2001), pp. 200–222. ISSN: 1741-2846. DOI: 10.1177/109434200101500302.

[31] Felix Gessert, Steffen Friedch, Wolfram Wingerath, Michael Schaarschmidt, and Norbert Ritter. "Towards a Scalable and Unified REST API for Cloud Data Stores." In: *44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014*. Ed. by Erhard Plödereder, Lars Grunske, Eric Schneider, and Dominik Ull. Bonn: Gesellschaft für Informatik, 2014, pp. 723–734. ISBN: 978-3-88579-626-8.

[32] David Giaretta. *Advanced digital preservation*. Springer Berlin, 2011. ISBN: 978-3-64216-808-6.

[33] Marco Giorgetta, Johann Jungclaus, Christian Reick, Stephanie Legutke, Victor Brovkin, Traute Crueger, Monika Esch, Kerstin Fieg, Ksenia Glushak, Veronika Gayler, Helmuth Haak, Heinz-Dieter Hollweg, Stefan Kinne, Luis Kornblueh, Daniela Matei, Thorsten Mauritsen, Uwe Mikolajewicz, Wolfgang Müller, Dirk Notz, Thomas Raddatz, Sebastian Rast, Erich Roeckner, Marc Salzmann, Hauke Schmidt, Reiner Schnur, Joachim Segschneider, Katharina Six, Martina Stockhause, Joerg Wegner, Heinrich Widmann, Karl-Hermann Wieners, Martin Claussen, Jochem Marotzke, and Bjorn Stevens. *CMIP5 simulations of the Max Planck Institute for Meteorology (MPI-M) based on the MPI-ESM-LR model: The abrupt4xCO2 experiment, served by ESGF*. World Data Center for Climate (WDCC), 2012. DOI: 10.1594/WDCC/CMIP5.MXELC2.

[34] Mike Graves, Adam Constabaris, and Dan Brickley. "FOAF: Connecting People on the Semantic Web." In: *Cataloging & Classification Quarterly* 43.3-4 (Apr. 2007), pp. 191–202. DOI: 10.1300/j104v43n03_10.

[35] Christophe Guéret, Paul Groth, Frank van Harmelen, and Stefan Schlobach. "Finding the Achilles Heel of the Web of Data: Using Network Analysis for Link-Recommendation." In: *The Semantic Web – ISWC 2010*. Ed. by Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm. Vol. 6496. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 289–304. DOI: 10.1007/978-3-642-17746-0_19.

[36] Paul R. Halmos. *Naive set theory*. Springer, 1974. ISBN: 978-0-38790-092-6.

[37] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, Feb. 2011. DOI: 10.2200/s00334ed1v01y201102wbe001.

[38] Tony Hey, Stewart Tansley, and Kristin Tolle. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research, 2009. ISBN: 9780982544204.

[39] Tony Hey and Anne E. Trefethen. "Cyberinfrastructure for e-Science." In: *Science* 308.5723 (May 2005), pp. 817–821. ISSN: 1095-9203. DOI: 10.1126/science.1110410.

[40] Sarah Higgins. "The DCC Curation Lifecycle Model." In: *International Journal of Digital Curation* 3.1 (Dec. 2008), pp. 134–140. ISSN: 1746-8256. DOI: 10.2218/ijdc.v3i1.48.

[41] Hans-Werner Hilse and Jochen Kothe. *Implementing persistent identifiers*. Consortium on European Research Libraries, 2006. ISBN: 9069845083.

[42] Nicole von der Hude. "Persistent Identifier: Versionierung, Addressierung und Referenzierung." In: *Langzeitarchivierung von Forschungsdaten*. Ed. by Reinhard Altenhöner and Claudia Oellers. Berlin: SCIVERO Verlag, 2012, pp. 129–135. ISBN: 978-3-944417-00-4.

[43] Denis Huschka, Claudia Oellers, Notburga Ott, and Gert G. Wagner. "Datenmanagement und Data Sharing. Erfahrungen in den Sozial- und Wirtschaftswissenschaften." In: *Handbuch Forschungsdatenmanagement*. Ed. by Stephan Büttner, Hans-Christoph Hobohm, and Lars Müller. Bad Honnef: Bock + Herchen, 2011. ISBN: 978-3-88347-283-6.

[44] Ronald Jantz and Michael J. Giarlo. "Digital Preservation: Architecture and Technology for Trusted Digital Repositories." In: *Microform & Imaging Review* 34.3 (Jan. 2005). ISSN: 0949-5770. DOI: 10.1515/mfir.2005.135.

[45] Robert Kahn and Robert Wilensky. "A framework for distributed digital object services." In: *International Journal on Digital Libraries* 6.2 (Apr. 2006), pp. 115–123. ISSN: 1432-5012. DOI: 10.1007/s00799-005-0128-x.

[46] Gary King. "Ensuring the Data-Rich Future of the Social Sciences." In: *Science* 331.6018 (Feb. 2011), pp. 719–721. ISSN: 1095-9203. DOI: 10.1126/science.1197872.

[47] Jochen Klar and Harry Enke. *Rahmenbedingungen einer disziplinübergreifenden Forschungsdateninfrastruktur – Report "Organisation und Struktur"*. Report. 2013. DOI: 10.2312/RADIESCHEN_005.

[48] Jens Klump, Roland Bertelmann, Jan Brase, Michael Diepenbroek, Hannes Grobe, Heinke Höck, Michael Lautenschlager, Uwe Schindler, Irina Sens, and Joachim Wächter. "Data publication in the open access initiative." In: *Data Science Journal* 5 (2006), pp. 79–83. DOI: 10.2481/dsj.5.79.

[49] W. Koehler. "A longitudinal study of Web pages continued: a consideration of document persistence." In: *Information Research* 9.2 (Jan. 2004).

[50] David Koop, Emanuele Santos, Phillip Mates, Huy T. Vo, Philippe Bonnet, Bela Bauer, Brigitte Surer, Matthias Troyer, Dean N. Williams, Joel E. Tohline, Juliana Freire, and Cláudio T. Silva. "A Provenance-Based Infrastructure to Support the Life Cycle of Executable Papers." In: *Procedia Computer Science* 4 (Jan. 2011), pp. 648–657. ISSN: 18770509. DOI: 10.1016/j.procs.2011.04.068.

[51] Michael Kuhn, Konstantinos Chasapis, Manuel F. Dolz, and Thomas Ludwig. "Compression by Default — Reducing Total Cost of Ownership of Storage Systems." In: *29th International Supercomputing Conference, ISC 2014*. Ed. by Julian M. Kunkel, Thomas Ludwig, and Hans W. Meurer. Vol. 8488. LNCS. Leipzig, Germany: Springer International Publishing, 2014, pp. 508–510. ISBN: 978-3-319-07517-4.

[52] John A. Kunze. "Towards Electronic Persistence Using ARK Identifiers, ARK motivation and overview." In: *Proceedings of the 3rd ECDL Workshop on Web Archives*. Trondheim, Norway, Aug. 2003.

[53] Massimo Lamanna. "The LHC computing grid project at CERN." In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 534.1-2 (Nov. 2004), pp. 1–6. ISSN: 01689002. DOI: 10.1016/j.nima.2004.07.049.

[54] Michael Lautenschlager. "Institutionalisierte "Data Curation Services"." In: *Handbuch Forschungsdatenmanagement*. Ed. by Stephan Büttner, Hans-Christoph Hobohm, and Lars Müller. Bad Honnef: Bock + Herchen, 2011, pp. 149–156. ISBN: 978-3-88347-283-6.

[55] Steve Lawrence, David M. Pennock, Gary W. Flake, Robert Krovetz, Frans M. Coetzee, Eric Glover, Finn Årup Nielsen, Andries Kruger, and C. Lee Giles. "Persistence of Web references in scientific research." In: *Computer* 34.2 (Feb. 2001), pp. 26–31. ISSN: 0018-9162. DOI: 10.1109/2.901164.

[56] Barbara Liskov and Stephen Zilles. "Programming with abstract data types." In: *Proceedings of the ACM SIGPLAN symposium on Very high level languages*. Vol. 9. 4. Santa Monica, California, USA: ACM, 1974, pp. 50–59. DOI: 10.1145/800233.807045.

[57] Barbara H. Liskov and Jeannette M. Wing. "A Behavioral Notion of Subtyping." In: *ACM Trans. Program. Lang. Syst.* 16.6 (Nov. 1994), pp. 1811–1841. ISSN: 0164-0925. DOI: 10.1145/197320.197383.

[58] Łukasz Bolikowski, Aleksander Nowiński, and Wojtek Sylwestrzak. "A System for Distributed Minting and Management of Persistent Identifiers." In: *10th International Digital Curation Conference*. London, UK, 2015.

[59] Peter Lyman. "Archiving the World Wide Web." In: *Building a National Strategy for Digital Preservation: Issues in Digital Media Archiving*. Washington, DC: Council on Library, Information Resources, and Library of Congress, Apr. 2002. ISBN: 1-887334-91-2.

[60] M. Mealling and R. Denenberg, eds. *RFC 3305: Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations*. IETF, 2002.

[61] Gerald A. Meehl, Richard Moss, Karl E. Taylor, Veronika Eyring, Ronald J. Stouffer, Sandrine Bony, and Bjorn Stevens. "Climate Model Intercomparisons: Preparing for the Next Phase." In: *Eos, Transactions American Geophysical Union* 95.9 (2014), pp. 77–78. ISSN: 00963941. DOI: 10.1002/2014eo090001.

[62] Paolo Missier, Bertram Ludäscher, Shawn Bowers, Saumen Dey, Anandarup Sarkar, Biva Shrestha, Ilkay Altintas, Manish K. Anand, and Carole Goble. "Linking multiple workflow provenance traces for interoperable collaborative science." In: *5th Workshop on Workflows in Support of Large-Scale Science (WORKS)*. IEEE, Nov. 2010, pp. 1–8. ISBN: 978-1-4244-8989-3. DOI: 10.1109/works.2010.5671861.

[63] R. Moats. *RFC 2141: URN Syntax*. IETF, 1997.

[64] Reagan Moore. "Towards a Theory of Digital Preservation." In: *International Journal of Digital Curation* 3.1 (Dec. 2008), pp. 63–75. ISSN: 1746-8256. DOI: 10.2218/ijdc.v3i1.42.

[65] Luc Moreau. "The Foundations for Provenance on the Web." In: *Foundations and Trends in Web Science* 2.2-3 (Feb. 2010), pp. 99–241. ISSN: 1555-077X. DOI: 10.1561/1800000010.

[66] Luc Moreau and Paolo Missier, eds. *PROV-DM: The PROV Data Model*. W3C Recommendation. World Wide Web Consortium, 2013.

[67] Luc Moreau, Bertram Ludäscher, Ilkay Altintas, Roger S. Barga, Shawn Bowers, Steven Callahan, George Chin, Ben Clifford, Shirley Cohen, Sarah Cohen-Boulakia, Susan Davidson, Ewa Deelman, Luciano Digiampietri, Ian Foster, Juliana Freire, James Frew, Joe Futrelle, Tara Gibson, Yolanda Gil, Carole Goble, Jennifer Golbeck, Paul Groth, David A. Holland, Sheng Jiang, Jihie Kim, David Koop, Ales Krenek, Timothy McPhillips, Gaurang Mehta, Simon Miles, Dominic Metzger, Steve Munroe, Jim Myers, Beth Plale, Norbert Podhorszki, Varun Ratnakar, Emanuele Santos, Carlos Scheidegger, Karen Schuchardt, Margo Seltzer, Yogesh L. Simmhan, Claudio Silva, Peter Slaughter, Eric Stephan, Robert Stevens, Daniele Turi, Huy Vo, Mike Wilde, Jun Zhao, and Yong Zhao. "Special Issue: The First Provenance Challenge." In: *Concurrency and Computation: Practice and Experience* 20.5 (2008), pp. 409–418. DOI: 10.1002/cpe.1233.

[68] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan V. den Bussche. "The Open Provenance Model core specification (v1.1)." In: *Future Generation Computer Systems* 27.6 (June 2011), pp. 743–756. ISSN: 0167739X. DOI: 10.1016/j.future.2010.07.005.

[69] Nick Nicholas, Nigel Ward, and Kerry Blinco. "A Policy Checklist for Enabling Persistence of Identifiers." In: *D-Lib Magazine* 15.1/2 (Jan. 2009). ISSN: 1082-9873. DOI: 10.1045/january2009-nicholas.

[70] *Open archival information system (OAIS) Reference model*. ISO Standard, ISO 14721:2012.

[71] N. Paskin. "Toward unique identifiers." In: *Proceedings of the IEEE* 87.7 (July 1999), pp. 1208–1227. ISSN: 0018-9219. DOI: 10.1109/5.771073.

[72] Norman Paskin. "Components of DRM Systems Identification and Metadata." In: *Digital Rights Management*. Ed. by Eberhard Becker, Willms Buhse, Dirk Günnewig, and Niels Rump. Vol. 2770. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 26–61. DOI: 10.1007/10941270_4.

[73] Norman Paskin. "Digital Object Identifier (DOI) System." In: *Encyclopedia of Library and Information Sciences, Third Edition*. Ed. by Marcia J. Bates and Mary N. Maack. Taylor & Francis, Dec. 2010. Chap. 157, pp. 1586–1592. DOI: 10.1081/e-elis3-120044418.

[74] Sandra Payette and Carl Lagoze. "Flexible and Extensible Digital Object and Repository Architecture (FEDORA)." In: *Research and Advanced Technology for Digital Libraries*. Vol. 1513. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 41–59. DOI: 10.1007/3-540-49653-x_4.

[75] Commission on Preservation and Access. *Preserving digital information: Report of the Task Force on Archiving of Digital Information*. 1996. ISBN: 1887334505.

[76] Arcot Rajasekar, Reagan Moore, Chien-Yi Hou, Christopher A. Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, Paul Tooby, and Bing Zhu. "iRODS Primer: Integrated Rule-Oriented Data System." In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2.1 (Jan. 2010), pp. 1–143. DOI: 10.2200/s00233ed1v01y200912icr012.

[77] Gunther Schmidt. *Relational mathematics*. Cambridge University Press, 2011. ISBN: 978-0-52176-268-7.

[78] Guus Schreiber and Yves Raimond, eds. *RDF 1.1 Primer*. W3C Recommendation. World Wide Web Consortium, 2014.

[79] European Commission High Level Expert Group on Scientific Data. *Riding the wave – How Europe can gain from the rising tide of scientific data*. European Union, Oct. 2010.

[80] Jamie Shiers. "The Worldwide LHC Computing Grid (worldwide LCG)." In: *Computer Physics Communications* 177.1-2 (July 2007), pp. 219–223. ISSN: 00104655. DOI: 10.1016/j.cpc.2007.02.021.

[81] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. "A Framework for Collecting Provenance in Data-Centric Scientific Workflows." In: *International Conference on Web Services (ICWS), 2006*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 427–436. ISBN: 0769526691. DOI: 10.1109/icws.2006.5.

[82] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. "A Survey of Data Provenance in e-Science." In: *ACM SIGMOD Record* 34.3 (Sept. 2005), pp. 31–36. ISSN: 0163-5808. DOI: 10.1145/1084805.1084812.

[83] Diomidis Spinellis. "The decay and failures of web references." In: *Communications of the ACM* 46.1 (Jan. 2003), pp. 71–77. ISSN: 0001-0782. DOI: 10.1145/602421.602422.

[84] Joan Starr and Angela Gastl. "isCitedBy: A Metadata Scheme for DataCite." In: *D-Lib Magazine* 17.1/2 (Jan. 2011). ISSN: 1082-9873. DOI: 10.1045/january2011-starr.

[85] M. Stockhause, H. Höck, F. Toussaint, and M. Lautenschlager. "Quality assessment concept of the World Data Center for Climate and its application to CMIP5 data." In: *Geoscientific Model Development* 5.4 (Aug. 2012), pp. 1023–1032. DOI: 10.5194/gmd-5-1023-2012.

[86] Heiko Stoermer. "OKKAM: Enabling entity-centric information integration in the semantic web." PhD thesis. University of Trento, Italy, 2008.

[87] S. Sun, S. Reilly, and L. Lannom. *RFC 3651: Handle System Namespace and Service Definition*. IETF, 2003.

[88] Karl E. Taylor, Ronald J. Stouffer, and Gerald A. Meehl. "An Overview of CMIP5 and the Experiment Design." In: *Bulletin of the American Meteorological Society* 93.4 (Oct. 2012), pp. 485–498. DOI: 10.1175/bams-d-11-00094.1.

[89] Kenneth Thibodeau. "Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years." In: *The State of Digital Preservation: An International Perspective*. Washington, DC: Council on Library and Information Resources, July 2002, pp. 4–31. ISBN: 1-887334-92-0.

[90] A. Treloar and C. Harboe-Ree. "Data management and the curation continuum: how the Monash experience is informing repository relationships." In: *VALA 2008*. Melbourne, Feb. 2008.

[91] Andrew Treloar. "The Research Data Alliance: globally coordinated action against barriers to data publishing and sharing." In: *Learned Publishing* (Sept. 2014), pp. 9–13. ISSN: 0953-1513. DOI: 10.1087/20140503.

[92] *Trusted Digital Repositories: Attributes and Responsibilities*. Technical report. Mountain View, CA, USA: Research Libraries Group, May 2002.

[93] Pieter Van Gorp and Steffen Mazanek. "SHARE: a web portal for creating and sharing executable research papers." In: *Procedia Computer Science* 4 (2011), pp. 589–597. ISSN: 18770509. DOI: 10.1016/j.procs.2011.04.062.

[94] Stuart L. Weibel and Erik Jul. "PURLs to improve access to Internet." In: *OCLC Newsletter* 218 (1995). Ed. by Nita Dean, George Promenschenkel, and Marifay Makssour. ISSN: 0163-898X.

[95] Tobias Weigel and Timothy DiLauro. "Separation of Concerns: PID Information Types and Domain Metadata." In: *International Conference on Dublin Core and Metadata Applications (DC-2013)*. Dublin Core Metadata Initiative. Lisbon, Portugal, 2013.

[96] Tobias Weigel, Timothy DiLauro, and Thomas Zastrow. *RDA Recommendation: PID Information Types*. Research Data Alliance, 2015. Under review.

[97] Tobias Weigel, Stephan Kindermann, and Michael Lautenschlager. "Actionable Persistent Identifier Collections." In: *Data Science Journal* 12 (2013), pp. 191–206. ISSN: 1683-1470. DOI: 10.2481/dsj.12-058.

[98] Tobias Weigel, Michael Lautenschlager, Frank Toussaint, and Stephan Kindermann. "A Framework for Extended Persistent Identification of Scientific Assets." In: *Data Science Journal* 12 (2013), pp. 10–22. ISSN: 1683-1470. DOI: 10.2481/dsj.12-036.

[99] Dean Williams, ed. *4th annual Earth System Grid Federation and Ultrascale Visualization Climate Data Analysis Tools face-to-face conference report*. Technical report LLNL-TR-666753. Livermore, CA, USA: Lawrence Livermore National Laboratory, 2014.

[100] Dean Williams, Gavin Bell, Luca Cinquini, Peter Fox, John Harney, and Robin Goldstone. "Earth System Grid Federation: Federated and Integrated Climate Data from Multiple Sources." In: *Earth System Modelling – Volume 6*. SpringerBriefs in Earth System Sciences. Springer Berlin Heidelberg, 2013, pp. 61–77. DOI: 10.1007/978-3-642-37244-5_7.

[101] Jonathan D. Wren. "URL decay in MEDLINE – a 4-year follow-up study." In: *Bioinformatics* 24.11 (June 2008), pp. 1381–1385. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btn127.

[102] Sarah J. Wright, Wendy A. Kozlowski, Dianne Dietrich, Huda J. Khan, Gail S. Steinhart, and Leslie McIntosh. "Using Data Curation Profiles to Design the Datastar Dataset Registry." In: *D-Lib Magazine* 19.7/8 (July 2013). ISSN: 1082-9873. DOI: 10.1045/july2013-wright.

[103] Elizabeth Yakel. "Digital curation." In: *OCLC Systems & Services* 23.4 (2007), pp. 335–340. ISSN: 1065-075X. DOI: 10.1108/10650750710831466.

EIDESSTATTLICHE VERSICHERUNG

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, Feb 27 2015

Tobias Weigel