

Ontology-based Software Architecture Documentation

Klaas Andries de Graaf

2015



SIKS Dissertation Series No. 2015-15

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This research has been partially sponsored by:

The Dutch “Regeling Kenniswerkers”, project KWR09164, “Stephenson: Architecture knowledge sharing practices in software product lines for print systems”.

The Natural Science Foundation of China (NSFC), project No. 61170025, “KeSRAD: Knowledge-enabled Software Requirements to Architecture Documentation”.

Promotiecommissie:

prof. dr. Rafael Capilla (King Juan Carlos University)

prof. dr. Paris Avgeriou (University of Groningen (RuG))

prof. dr. Dick Bulterman (VU University Amsterdam,
Centrum Wiskunde & Informatica)

prof. dr. Patricia Lago (VU University Amsterdam)

dr. Remco de Boer (ArchiXL)

ISBN 978-94-6295-145-7

Copyright © 2015, Klaas Andries de Graaf

All rights reserved unless otherwise stated.

Cover design and typeset in L^AT_EX by by author

Cover illustration ‘*Boekdrukkunst*’ ca. 1589 - ca. 1593, printmaking and publishing by Philips Galle, Antwerp. Based on design of Jan van der Straet. Source: Rijksmuseum, Amsterdam

Printed and published by Proefschriftmaken.nl || Uitgeverij BOXPress, 's-Hertogenbosch

VRIJE UNIVERSITEIT

Ontology-based Software Architecture Documentation

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. F.A van der Duyn Schouten,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Exacte Wetenschappen
op maandag 11 mei 2015 om 15.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Klaas Andries de Graaf

geboren te Middelburg

promotor: prof.dr. J.C. van Vliet
copromotoren: dr. A. Tang
dr. P. Liang

Acknowledgements

I want to express my gratitude towards the people involved in my doctoral studies. I am grateful for the guidance by my advisors; Hans van Vliet, Antony Tang, and Peng Liang. Hans, thanks for your continuing support, guidance, book recommendations, and for teaching me how to think and write more clearly. Antony, thanks for your friendly advice, teachings, Skype calls, and the visits across the world. Peng, thanks for the good discussions, teachings, many improvements, and our talks about research, language, and history.

Thanks to several of my colleagues at VU University Amsterdam; Patricia Lago, Christina Manteli, Damian Andrew Tamburri, Maryam Razavian, Giuseppe Proccianti, Han van der Aa, Rahul Premraj, Nelly Condori-Fernandez, and Rinke Hoekstra. Especially thanks to Patricia for her support and advice before, during, and after my PhD. Thanks Christina, for your help as well as the gezelligheid in the office and during TA duty. Thanks Damian, for the XP and quests. Thanks Maryam, for the gezelligheid, order, and plants in the office. Also thanks to Willem van Hage for the successful collaboration. Thanks to Elly Lammers, Caroline Waij, Kris de Jong, and Mojca Lovrencak for the nice talks and for helping me with the necessary procedures. Thanks to the friendly people from computer support and the FEW helpdesk. Thanks to the friendly people at Swinburne University of Technology for my pleasant stay there.

Many thanks to René Laan, Wim Couwenberg, John Kessler, Pieter Verduin, Amar Kalloe, and the other good folks at Océ R&D for their support, interest, participation, and excellent insights which contributed to this thesis. Many thanks to the good folks at LaiAn that contributed to the research in this thesis. Thanks to Jonathan Rebel, Ruben Hartog, and Berend van Veenendaal for their adaptations to OntoWiki. Thanks to the students that participated in the experiment during the 2012 Software Architecture course at the University of Amsterdam. Thanks to the reading committee, anonymous reviewers, and editors for evaluating this thesis or one of the papers used in this thesis.

Thanks to my parents, Roelof and Marrie, for supporting my studies and giving me early access to computers, software, books, bibles, work, museums, etc. Thanks to Gerrit for his advice on pursuing a PhD. Thanks to Willem, Hans, and Bram for coping with my absent mind in the past 4 years and for not harassing me too much.

Contents

1	Introduction	1
1.1	Software Development and Documentation	1
1.2	Software Architecture Documentation	2
1.3	Research Motivation	3
1.4	Research Questions	5
1.5	Research Approach	6
1.6	Thesis Chapters	10
1.7	Publications	12
2	Searching Architectural Knowledge in File-based Documenta- tion	15
2.1	Introduction	15
2.2	Design and Analysis of Search Behaviour Study	17
2.3	Using Prior Knowledge to Search under Uncertainty	25
2.4	Lessons Learnt	32
2.5	Threats to Validity	34
2.6	Related Work	35
2.7	Conclusions	36
3	Organising and Retrieving Architectural Knowledge in File-based Documentation	39
3.1	File-Based Documentation and its Issues	39
3.2	Hypertext Documentation and Its Issues	42
3.3	Conclusion	43
4	Ontology-based Architecture Documentation Approach	45
4.1	Software Architecture Ontologies	45
4.2	ArchiMind Semantic Wiki	48
4.3	Annotating SA Documentation in ArchiMind	51
4.4	Related Work	54
4.5	Conclusion	55
5	An Exploratory Study on Ontology Engineering for Architecture Documentation	57
5.1	Introduction	57
5.2	Background	59
5.3	Ontology Engineering using the 'Typical Question' Approach	61
5.4	Contextual Factors in Ontology Engineering	67
5.5	Case Study	72

CONTENTS

5.6	Related Work	78
5.7	Conclusions and Future Work	80
6	How Organisation of Architecture Documentation Influences Knowledge Retrieval	83
6.1	AK Retrieval Efficiency and Effectiveness	85
6.2	How AK Organisation Affects AK Retrieval	97
6.3	Qualitative Evaluation	107
6.4	Cost-Benefit Analysis	111
6.5	Threats to Validity	113
6.6	Implications	116
6.7	Related Work	118
6.8	Conclusions	118
7	Supporting Architecture Documentation: A Comparison of Ontologies for Knowledge Retrieval	121
7.1	Introduction	121
7.2	AK Retrieval Experiment	123
7.3	AK Organisation and AK Retrieval	132
7.4	Discussion	137
7.5	Threats to Validity	138
7.6	Conclusions	139
8	Conclusions	141
8.1	Contributions	141
8.2	Innovative Aspects	144
8.3	Discussion and Future Work	144
9	Samenvatting	149
	SIKS Dissertatiereeks	155
	Bibliography	186
	Abbreviations	187



Introduction

Software has enabled progress in many fields and is increasingly affecting our lives and society as a whole. Software is used in computers, communication networks, medical devices, factories, planes, trains, cars, mobile phones, household appliances, etcetera. Software systems that are badly designed, built, or maintained can malfunction and become slow, unsafe, and unreliable, which in turn results in the loss of information, time, money, or lives.

It is important that software systems operate as intended, however, software development is not trivial. Developing a single software system may require years of work by hundreds of professionals on several million lines of programming code. In this thesis we investigate whether we can improve the retrieval of knowledge from documentation that professionals use during software development.

1.1 Software Development and Documentation

The development of large software systems involves multiple software professionals that work together within a project. Software development activities in projects are planned in phases, e.g., in a requirement, design, and implementation phase, and in iterative cycles. Systematic development of software is part of software engineering, which can be defined as "*the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software*" [1], i.e., an engineering approach to software development.

The concept of software engineering was first introduced and discussed during the 1968 NATO software engineering Conference [71]. The increasing need for software, demands on its operation, and problems in developing large systems

required an evaluation of existing software development practices and education. Several conference attendees stressed that good documentation led to better software, with fewer errors and better design, and was invaluable for maintenance.

Software projects have documented deliverables such as requirement and design specifications. The documentation is used for communicating knowledge among professionals, especially if they work in different project phases, locations, and across different time zones. Professionals retrieve documented knowledge in order to co-develop a software system. Even in Agile development, where working software is valued over comprehensive documentation, practitioners regard documentation as important for their tasks and experience a lack of documentation [92].

It is generally agreed that documentation is essential for software development. Bad documentation causes inefficiency and errors throughout the development life-cycle [77], however, there is still much room for improving documentation practices [117]. Parnas argues [77] that software documentation is a "*perpetually unpopular topic*", and that software professionals do not write precise documentation compared to engineers in other disciplines.

1.2 Software Architecture Documentation

An early activity in the system development life-cycle is to specify the Software Architecture (SA) of a system. The SA of a system can be defined as "*the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both*" [9]. In an SA design the system is decomposed into interacting components to realize functional and non-functional requirements (e.g., performance and reliability), constraints of the technical and organisational environment, work breakdown, budget, planning, component reuse, and families of systems [116].

Documentation of SA serves three important purposes: it is used for system analysis, education, and it is the primary vehicle for communication between stakeholders in a project [21]. Architectural Knowledge (AK) is contained in SA documentation. AK can be defined as "*the integrated representation of the software architecture of a software-intensive system (or a family of systems), the architectural design decisions, and the external context/environment*" [64].

SA documentation is not only used in an early design stage or cycle, but guides the software development throughout a project. SA documentation is frequently revisited during software enhancement and maintenance [101]. Many practition-

ers forget the reasons behind architectural design decisions and do not understand the design of others without documented design rationale [101].

1.3 Research Motivation

It is recognized by Bass et al. in [9] and Clements et al. in [21] that even a perfect SA is essentially useless if it is not understood; proper documentation should have enough detail, no ambiguity, and it must be organized such that users can quickly find information and answer their questions [77]. In software industry, it is common practice to capture AK in file-based documents [80] such as text files and diagrams.

Parnas and Clements argue [78] that documents should be designed and organised with separation of concerns in mind; each aspect of a system is described in one section. A file-based document can be separated by concerns using, e.g., a view-based organisation [9, 21] in which each view describes an aspect of AK for interested document users. Separation of document content into sections provides an organisation of AK, and a table of content with section titles can be used as an index to find the AK in this organisation.

Many relationships exist between AK in SA documentation, e.g., between requirements, decisions, and components. Consider a decision recorded in a document. A developer may need to know how this decision impacts the components and interfaces s/he is working on. When evaluating the decision an architect is interested in related decisions, requirements, and alternatives. A quality assurance manager might need to know all quality attributes that the decision impacts.

Explicit documentation of the relationships between AK is also referred to as 'traceability' between AK. The importance of traceability between AK was recognized several decades ago [72], however, it is still difficult to achieve traceability between AK [49, 20]. Industry professionals indicate that the lack of traceability in SA documentation is a major problem [80].

In the book 'Documenting Software Architectures' [21], the first rule for sound SA documentation is that it should be written for its readers. This rule entails that multiple relationships between AK are described for the readers, since SA cannot be described in one dimension [21]. The organisation of file-based documents in a table of content is however linear, and introduces repeated descriptions of AK when multiple relationships between (or '*dimensions on*') the AK have to be found in this organisation. The repetition of AK in file-based documents however conflicts with the second rule for sound documentation in [21], which states that

CHAPTER 1. INTRODUCTION

AK should not be repeated, especially considering the difficulties of document maintenance when SA evolves.

Relationships between AK that are not indexed by the file-based document organisation have to be searched inside document contents within sections. However, reading or keyword searching in document contents can take much time and is error-prone due to synonyms, spelling errors, and abbreviations. Moreover, it is difficult to make document contents unambiguous [77] and organise the AK therein such that it is successfully communicated to users with different backgrounds [83].

It is hard to organise interrelated AK using the linear file-based document organisation in such a way that it supports all document users in finding the AK they need. SA documentation in industry is predominantly file-based and often has a 'one-size-fits-all' organisation that does not serve specific users and their tasks well [80]. SA documentation that is not suitable for its users is not cost-effective [21, 31].

A document organisation that does not support all AK needs may cause its users to waste time searching for AK in the wrong locations and retrieve incomplete or incorrect AK, which in turn leads to delays and mistakes during their software development activities. The inefficient and ineffective retrieval of AK from SA documentation is the problem we address in this thesis.

We conjecture that an ontology-based documentation approach can improve the retrieval of AK from SA documentation, compared to the file-based approach. "*An ontology*" refers to a formal domain model in which concepts and relationships between concepts are described [68]. An ontology can organise AK with classes and relationships, and provides explicit semantics for readers to recognize AK and relationships between AK. An ontology-based AK organisation is non-linear, and may help document users to retrieve the interrelated AK that they need more quickly and correctly.

There is a growing interest in the use of ontologies for SA documentation, e.g., for SA document re-use [119] and as a precise and common vocabulary for describing architectural decisions [4, 60]. Su *et al.* propose the use of ontology for visualisation and non-linear navigation of SA documentation to reduce the cognitive load on its users [95]. López *et al* [68] and Jansen *et al.* [51] provide empirical evidence that the use of ontology-based documentation improves AK extraction and AK understanding, respectively. We want to investigate if the use of ontology-based documentation improves the retrieval of AK needed by software professionals, and we want to explain why AK retrieval is (not) improved.

1.4 Research Questions

In this thesis we investigate the use of ontology-based documentation to improve the efficiency and effectiveness of AK retrieval from documentation. We study AK retrieval efficiency by measuring the time required to answer questions about AK, and effectiveness by measuring the correctness and completeness, i.e., precision and recall, of answers to the questions.

Our main Research Question (RQ) is:

Can we improve AK retrieval efficiency and effectiveness using ontology-based documentation?

We first need to find out how AK is retrieved in file-based documentation, to understand how ontology-based documentation may improve AK retrieval. We study practice and literature to investigate how professionals find file-based AK descriptions and use AK organisation, and to identify AK retrieval challenges. Our first RQ is thus:

RQ1 *How do software professionals retrieve AK from file-based documentation?*

Next, we examine how an ontology can be used for AK retrieval from SA documentation. We introduce an ontology-based approach for storing, annotating, and retrieving AK descriptions together with their lay-out, diagrams, and the meta-data of SA documents. Our second RQ is:

RQ2 *How can an ontology be used for retrieving AK from documentation?*

Building an ontology for SA documentation requires ontology engineering, and we investigate how a useful ontology may be built in the context of a software industry project. Different roles in software development have different needs for AK, and these needs may be complex and domain specific. Building an ontology to suit these diverse AK needs is important, since professionals need to retrieve the AK from documentation, and challenging, because professionals in industry have limited time and opportunity to provide and clarify their AK needs. Hence, our third RQ is:

RQ3 *How to construct an ontology for SA documentation in a software project context?*

To ascertain that the ontology-based approach improves AK retrieval, we compare it to a file-based approach in software industry practice. We test whether there is a significant difference in AK retrieval efficiency and effectiveness between software professionals that answer architecture-related questions from file-based

and ontology-based documentation. We want to explain why there is (no) difference in AK retrieval efficiency and effectiveness and understand how the use of the approaches influences AK retrieval by analysing the recorded search actions. This leads to our fourth RQ:

RQ4 *How do file-based and ontology-based documentation influence the efficiency and effectiveness of AK retrieval?*

Finally, we want to understand how different ontologies perform in terms of their relative efficiency and effectiveness, in order to optimize AK retrieval of the ontology-based approach itself. We test for differences in AK retrieval efficiency and effectiveness between the use of ontologies built from different understandings of the AK needs of document users. We analyse the search actions of document users in order to understand how the use of different ontology-based AK organisations influences AK retrieval. Our fifth RQ is:

RQ5 *How do different ontology-based AK organisations influence the efficiency and effectiveness of AK retrieval?*

1.5 Research Approach

The five research questions together provide insights that are used to answer the main RQ. Figure 1.1 depicts how the RQs relate to each other via the objects that are studied.

File-based SA documentation is studied in RQ1 to understand how professionals retrieve its AK. Insights from RQ1 are input for RQ2 to find out how an ontology can be used to retrieve AK. The AK descriptions in file-based documentation are imported in ontology-based documentation and both approaches are compared when investigating RQ4.

Ontologies organise the AK in ontology-based documentation. We constructed an ontology in a software project context (RQ3). The newly constructed ontology and a predefined ontology were used in ontology-based documentation that was compared with file-based documentation (RQ4). Two other ontologies were constructed and their use in ontology-based documentation was compared to investigate the effect of different ontology-based AK organisations on AK retrieval efficiency and effectiveness (RQ5).

An ontology-based documentation approach was created as a result of RQ2. The approach was compared to file-based documentation (RQ4) and optimized by comparing different ontologies (RQ5). RQ4 and RQ5 provide insights for our

1.5. RESEARCH APPROACH

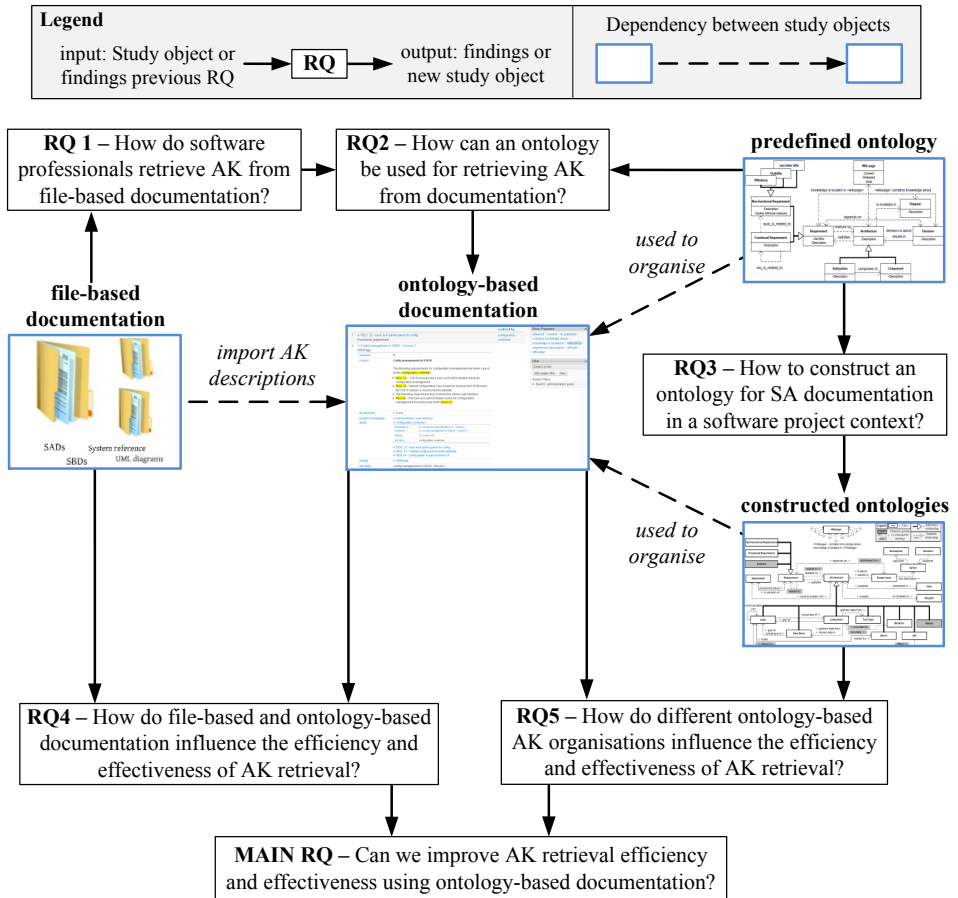


Figure 1.1: Relationships between research questions and study objects

main RQ; whether AK retrieval efficiency and effectiveness can be improved using ontology-based documentation.

We applied several research methods to answer the RQs:

Controlled Experiments are used to test hypotheses about the effect of independent (or 'predictor') variables on dependent (or 'response') variables. There is control on the experiment context to limit the effects of variables other than the chosen independent variables [34]. We tested hypotheses about the effect of using file-based and ontology-based documentation (independent variables) on the efficiency and effectiveness (dependent variables) of AK retrieval.

CHAPTER 1. INTRODUCTION

A *Case study* is an empirical investigation for which the control and reductionism in an experiment is not suitable [34]. Case studies are appropriate to investigate a "*phenomenon in depth within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident*" [120]. We conducted an exploratory case study [82] to gain insight in the process of building an ontology in the context of a software project.

Protocol Analysis [35] is the study of verbal reports to identify how people use their intelligence to solve problems in complex real world environments [19]. Analysis of verbal reports can be used to identify cognitive processes and build knowledge-based systems [115]. We asked software professionals to voice their thoughts whilst searching in SA documents and analysed the transcripts.

Grounded Theory (GT) uses empirical generalization to build a domain theory [40] around a central theme [93]. Patterns that indicate concepts are identified in collected domain data. The identified concepts are aggregated into categories, relationships between categories, and their properties, which together form a domain theory. We used GT to build an ontology (i.e., a domain theory) around the central theme; "*AK that needs to be retrieved from SA documentation*".

A *Survey* is used to collect both objective data (e.g., demographics) and subjective data (e.g., opinions) from individuals via interviews or questionnaires [55]. The data collected from a representative sample of a population may be generalized to identify characteristics of the total population [34].

We conducted *Literature review* by studying books, publications found via internet search engines, publications found in bibliographies, and publications that cited the literature we previously studied. This is different from a *systematic* literature review, which has the goal of sampling and aggregating evidence in literature (considered primary studies) and reporting the review results as a complete (secondary) study [12].

During *Prototyping* an initial system version with its most essential functions is built. Prototyping allows exploration of design issues as well as early communication of the functionality and design principles of a system [113].

The Computer Research Methods (CRM) framework by Holz *et al.* [50] can be used to describe a study with four questions:

1. Research objective: *what do we want to achieve?*
2. Data collection: *where does our data come from?*
3. Data usage: *what do we do with the data?*
4. Achievement: *was the goal achieved?*

1.5. RESEARCH APPROACH

We summarize the research conducted for each RQ below using the first three CRM questions. Our conclusion chapter answers the fourth CRM question. Answers to CRM questions¹ and the applied research methods are listed in italic between parentheses. Table 1.1 gives an overview of the aforementioned.

Table 1.1: Overview of research objective, data collection, data usage, and research methods per RQ

Research questions	Research objective	Data collection	Data usage	research methods
RQ1 in Chapter 3 and 2	understand technology	observe and measure in field, read	identify patterns, themes, and trends	protocol analysis, literature review
RQ2 in Chapter 4	create technology	model and implement	develop technology	prototyping
RQ3 in Chapter 5	create approach and understand	ask and model in field	develop approach, identify trends	case study, grounded theory
RQ4 in Chapter 6	compare, evaluate, and understand technology	experiment, measure, and ask in field	calculate numbers, identify patterns	controlled experiment, survey
RQ5 in Chapter 7	compare, evaluate, and understand technology	model, experiment, and measure	calculate numbers, identify patterns	controlled experiment

The objective of RQ1 is to understand how software professionals retrieve AK from file-based documentation (objective: *understand technology*). We collected data by recording the actions of software professionals that search for AK in file-based documentation (data collection: *observe and measure in field*) and by reviewing literature (data collection: *read*, method: *literature review*). We used the collected data to investigate the cognitive process of professionals that search for AK and to identify how AK is typically organised and retrieved in file-based documentation (data usage: *identify patterns, themes, and trends*, method: *protocol analysis*).

The objective of RQ2 is to create an ontology-based approach for retrieving AK from SA documentation (objective: *create technology*). We specified and implemented an ontology-based documentation approach (data collection: *model and implement*, data usage: *develop technology*, method: *prototyping*).

¹We use "understand" and "create" as shorter terms for the research objectives in [50] and added "develop" to the data usage sources in [50].

The objective of RQ3 is to understand how we can construct an ontology for SA documentation in a software project context (objective: *create approach and understand*). We collected data in a cyclic process, by first acquiring typical questions from software professionals (data collection: *ask in field*, method: *case study*), using the questions to model an ontology (data collection: *model in field*, method: *grounded theory*), and evaluating the ontology with the help of other professionals (data collection: *ask in field*). This process was repeated, evaluated, and specified as an ontology engineering approach (data usage: *develop approach and identify trends*).

The objective of RQ4 is to understand how the use of a file-based and ontology-based documentation approach influences AK retrieval efficiency and effectiveness (objective: *compare, evaluate, and understand technology*). We collected data in an experiment involving software professionals that answered questions about AK using the two documentation approaches (data collection: *experiment and measure in field*, method: *controlled experiment*), and by conducting a survey among professionals (data collection *ask in field*, method: *survey*). The collected data was used to test for a significant difference in AK retrieval efficiency and effectiveness between the approaches. We analysed the search actions of experiment participants to explain how the two approaches influenced AK retrieval (data usage: *calculate numbers, identify patterns*).

The objective of RQ5 is to understand how ontology-based AK organisation can be optimized for efficient and effective AK retrieval (objective: *compare, evaluate, and understand technology*). We built two ontology-based AK organisations that were used to answer questions during architectural review (data collection: *model, experiment, and measure*, method: *controlled experiment*). The collected data was used to test for a significant difference in AK retrieval efficiency and effectiveness between the two ontology-based AK organisations. We analysed the search actions of experiment participants to explain how the AK organisations influenced AK retrieval (data usage: *calculate numbers, identify trends*).

1.6 Thesis Chapters

- **Chapter 2 - Searching Architectural Knowledge in File-based Documentation**

In this chapter we examine how software professionals retrieve AK from file-based documentation (**RQ1**). We captured the search actions of software professionals that use file-based SA documentation in an industry case study and we investigated their cognitive process using protocol analysis. We found that prior knowl-

edge helps professionals to search AK efficiently and effectively. However, it can also misguide professionals to an incomplete search.

- **Chapter 3 - Organising and Retrieving Architectural Knowledge in File-based Documentation.**

In this chapter we review literature to find out how AK is typically retrieved from file-based documentation (**RQ1**). We found that file-based documents have a linear organisation of AK whilst document users do not necessarily retrieve AK following the same organisation. Users may not easily find AK that not indexed by the document organisation, however, creating a document organisation that supports the AK retrieval needs of all users is difficult and introduces redundant and scattered AK descriptions. AK retrieval challenges reported in literature stem from limitations of the linear file-based document organisation.

- **Chapter 4 - Ontology-based Architecture Documentation Approach**

In this chapter we investigate how an ontology can be used for retrieving AK from SA documentation (**RQ2**). We first give background information on the use of ontologies for organising and retrieving AK. We then introduce an ontology-based documentation approach that consists of a software ontology and semantic wiki.

- **Chapter 5 - An Exploratory Study on Ontology Engineering for Architecture Documentation**

This chapter illustrates how to build an ontology for SA documentation in a software project (**RQ3**). Different roles in software development have different needs for AK, and building an ontology to suit these diverse needs is challenging. We describe an approach that involves the use of typical questions and grounded theory for eliciting and constructing an ontology. We outline eight contextual factors, which influence the successful construction of an ontology, especially in complex software projects with diverse AK users. We tested our 'typical question' approach in an industrial case study and report how it can be used for acquiring and modelling AK needs to construct a useful ontology.

- **Chapter 6 - How Organisation of Architecture Documentation Influences Knowledge Retrieval**

In this chapter we report case studies in two companies to investigate how the use of file-based and ontology-based documentation influences AK retrieval (**RQ4**). We tested if there was a difference in AK retrieval efficiency and effectiveness between software professionals that answered architecture-related questions from the two documentation approaches in a controlled experiment. We then investigated why there was (no) difference in AK retrieval efficiency and effectiveness between the approaches by studying the search actions of the software profes-

sionals. We found that the use of better AK organisation correlates with the efficiency and effectiveness of AK retrieval. We also conducted surveys and a cost-benefit analysis of adopting ontology-based documentation in the studied projects.

• Chapter 7 - Supporting Architecture Documentation: A Comparison of Ontologies for Knowledge Retrieval

In this chapter, we investigate how different AK organisations influence the efficiency and effectiveness of AK retrieval from ontology-based documentation (**RQ5**). We executed a controlled experiment to test for differences in AK retrieval efficiency and effectiveness between ontologies built from different understandings of the AK needs of document users. We found that an improved understanding of AK needs allows for the construction of an ontology from which document users retrieve AK more efficiently and effectively. In constructing the ontologies, we applied ontology design criteria suggested by Gruber [42] to improve their general qualities. In some cases we found that the ontology support for AK needs had to be traded off against ontology design criteria.

• Chapter 8 - Conclusions

In this chapter we summarize and discuss the answers to RQ1 through RQ5, and how they together provide an answer to the **main RQ**. We describe contributions of the work in this thesis as well as their implications and possible future work.

1.7 Publications

Most writings in this thesis are peer-reviewed publications or currently under review for publication.

Chapter 2 was published as:

- K. A. de Graaf, P. Liang, A. Tang, and H. van Vliet - "The impact of prior knowledge on searching in software documentation", In *ACM Symposium on Document Engineering (DocEng)*, pp. 189-198, 2014. [27]

Chapter 3 is a section from an article that is under review for publication as:

- K. A. de Graaf, P. Liang, A. Tang, and H. van Vliet - "How organisation of architecture documentation affects architectural knowledge retrieval", *Science of Computer Programming (SCP) - Special Issue on Knowledge-based Software Engineering, March 2016 - under review*. [29].

We refer to this article as "*SCP article*". The SCP article is an extension of:

- K. A. de Graaf, A. Tang, P. Liang, and H. van Vliet - "Ontology-based software architecture documentation", In *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 121-130. IEEE, 2012. [30]

Chapter 4 is based on the SCP article [29] (under review) except for Section 4.3, which was published as:

- K. A. de Graaf - "Annotating software documentation in semantic wikis", In *Workshop on Exploiting semantic annotations in information retrieval (ESAIR)*, pp. 5-6. ACM, 2011. [25]

Chapter 5 was published as:

- K. A. de Graaf, P. Liang, A. Tang, W. R. van Hage, and H. van Vliet - "An exploratory study on ontology engineering for software architecture documentation", *Computers in Industry*, 65(7):1053-1064, 2014. [26]

The writings in chapter 6 are from the SCP article [29] that is under review and extended from [30].

Chapter 7 will be published as:

- K. A. de Graaf, P. Liang, A. Tang, and H. van Vliet - "Supporting architecture documentation: A comparison of two ontologies for knowledge retrieval", In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2015. [28] (see <http://dx.doi.org/10.1145/2745802.2745804>)

2

Searching Architectural Knowledge in File-based Documentation

In this chapter we examine how software professionals retrieve AK in file-based documentation (RQ1). It is important that AK can be retrieved efficiently and effectively, to prevent wasted time and errors that negatively affect the quality of software. We studied the search behaviour of professionals in industry when they answered questions using SA documents. Prior knowledge helps professionals to search SA documents efficiently and effectively. However, it can also misguide professionals to an incomplete search¹.

2.1 Introduction

In software industry, it is a common practice to capture information about a software system, its design, and architecture in file-based documents [80], e.g., in text documents and diagram files. It is important that software professionals can quickly and correctly answer questions from these documents. Otherwise valuable time is wasted, costly errors could be made, and software may not be built according to specification, which increases the cost of software projects and decreases the quality of software.

The organisation of file-based documents by directories, titles, and sections typically does not support all of the questions asked by software professionals [77]. Spelling errors, abbreviations, and synonyms make keyword searching ineffective and professionals may not know the right keywords to find answers [65]. Exhaust-

¹In this study context we use 'AK retrieval' and 'AK searching' interchangeably.

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

tive exploration of all document content is time-consuming and impractical in a large document set.

These issues introduce search uncertainty and make it hard for professionals to find complete and correct answers within reasonable time. Professionals waste time searching for answers in unstructured documentation [77]. The obstacles to finding the right information can be so great that it discourages professionals from trying to search at all [66, 65].

We studied how 26 software professionals in industry retrieved AK from documentation to answer architecture-related questions. The software professionals were asked to think aloud while answering questions about software and architectural elements such as subsystems, components, behaviour, requirements, and decisions. We measured how much time was spent on finding answers to the questions and whether answers were complete and correct.

We found that the search behaviour of software professionals is heavily influenced by their prior knowledge about the documentation and the software specified in this documentation. Prior knowledge is used to guide predictions about, e.g., the location of knowledge, which keywords can be used to find knowledge, and whether the knowledge found is correct and complete. Professionals use their prior knowledge as a short-cut to find answers to their questions, i.e. they use a heuristic (or 'experience-based') approach [108, 91] to searching.

Use of prior knowledge helped some of the participants in the study to quickly find the location of correct answers, even when the document organisation did not support the questions asked. The participants preferred to use their prior knowledge instead of exhaustively exploring documentation content.

We however observed that availability and confirmation bias can occur when using prior knowledge, which results in wasted time and incomplete answers. Availability bias and confirmation bias are cognitive biases that cause errors in judgement. Participants made inaccurate predictions about whether documents contained answers and whether searching for certain keywords would lead to answers. Moreover, several participants only looked for confirmation of answers that they said to know from their prior knowledge.

In this chapter we first describe how prior knowledge is used by professionals to search AK in SA documentation. We then evaluate the use of prior knowledge in terms of AK retrieval efficiency and effectiveness and report cognitive biases that lower this efficiency and effectiveness. These findings provide guidance for software practitioners to make optimal use of their prior knowledge when searching AK in SA documentation.

2.2. DESIGN AND ANALYSIS OF SEARCH BEHAVIOUR STUDY

We make the following contributions:

1. Report how professionals use prior knowledge to search in SA documents.
2. Identify cognitive biases that may occur when using prior knowledge to search in SA documents.
3. Report how prior knowledge and cognitive bias affect the efficiency and effectiveness of searching.

Section 2.2 details on the study design, identification of the search strategies, and cognitive process of participants. Section 2.3 reports and evaluates how prior knowledge is used when applying the search strategies and how cognitive biases may occur. Lessons learnt for document users and writers are described in Section 2.4 and Section 2.5 discusses threats to validity. In Section 2.6 we discuss related work and Section 2.7 reports our conclusions.

2.2 Design and Analysis of Search Behaviour Study

2.2.1 Study Design

We conducted a study to investigate how software professionals search for AK in SA documents. This study was part of a larger experiment reported in Chapter 6. The study was conducted in a software project at the R&D department of Océ technologies in the Netherlands. Océ is an international leader in digital document management and a Canon Group company. Océ applies an agile development methodology to encourage creativity and productivity.

Participants are all software professionals at Océ R&D who are involved in the software development process. Océ participants were recruited by circulating a voluntary sign-up list during a presentation about ArchiMind (advertised using a mailing list and posters). At the end of each experiment session we asked participants to recommend interested colleagues. This is a form of snowball sampling. Table 2.1 gives the demographics of the participants.

The Océ professionals need to retrieve AK specified in the reference architecture for a product-line of printing machines which also details on the variations and configuration of specific products. With the help of an Océ professional we estimated that in 7 months time at least 49 out of 145 product-line architecture documents were actively used in multiple projects. The documents used in this study are a subset of all reference architecture documentation.

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

Table 2.1: Demographics of Participants at Océ

Number of participants	Primary role of participants	Average years in role at Océ	Average years in role	Average years working at Océ
6	Domain architect	3.60	4.77	9.92
5	Software engineer	6.47	6.81	7.47
5	Software project manager	3.83	5	14
4	Product or system test engineer	9.75	11.75	11.625
4	Workflow architect	7.25	7.25	18.75
1	Configuration manager	3	10	3
1	Software designer	1	1	1

The documents used in the Océ study are:

- Two Software Architecture Documents (SAD) of 3 and 9 pages. SADs detail the design of functionality, behaviour, and components. One SAD gives an overview of the AK in the other SAD.
- Four Software Behaviour Documents (SBD), ranging in size from 8 to 18 pages. SBDs describe the behaviour of software together with all requirements and settings for that behaviour.
- One System Reference Document (Sysref) of 19 pages. The Sysref details on the high level system design, its decomposition in terms of subsystems, components, and interfaces, and decisions and rationale on the system design.
- One Design Document containing three UML diagrams that detail on the design of subsystems, components, and interfaces. The design document is often more up to date than the Sysref document that partially details on the same AK.

These documents follow a company-specific format and do not mention usage of certain architecture description standards, e.g., ISO 42010 [2]. The documents are stored in 3 directories. A directory 'Sysref' contains the Sysref and the design document in UML. A directory 'SBD' contains SBDs. A directory SAD contains the overview SAD and one subdirectory with the other SAD.

The documents are written in English, and consist of 79 pages, 3 diagrams, 1,794 paragraphs, 3,183 lines, and 13,962 words. Participants could search the documents using a file explorer (MS Windows Explorer), document editor (MS

2.2. DESIGN AND ANALYSIS OF SEARCH BEHAVIOUR STUDY

Word), and UML editing tool (MagicDraw).

An Océ professional estimated that there are around 50-75 users of these documents. Three Océ software professionals confirmed that the documents are representative of their usual practice. Question 6 of a questionnaire among participants in Table 6.4 in Section 6.3 also confirms this.

We formulated 7 questions about the knowledge in the documents. Criterion for selection of these questions include that the interpretation of the questions is similar between different participants and that their answers can be quantitatively assessed, i.e., the questions should not be open-ended. Part of the questions have been obfuscated for non-disclosure reasons: ‘QQ’, ‘XX’, ‘YY’, and ‘ZZ’ replace an actual software entity or concept.

1A: *Which settings have an impact on behaviour “History”?*

1B: *Which settings have an impact on behaviour “Alert Light”?*

2: *Which requirements for behaviour “XX” should be satisfied (realized) by component “Settings Editor”?*

3A: *Which decisions have been made about component “Settings Editor”?*

3B: *Which decisions have been made on the configuration of behaviour “YY”, “ZZ”, “History”, and “XX”?*

4A: *Which subsystem is interface “QQ” part of?*

4B: *Which other interfaces are offered by this subsystem?*

13 of the 26 participants answered questions 1A, 1B, and 2 and the other 13 participants answered questions 3A, 3B, 4A, and 4B. Answering these questions was part of an experiment reported in more detail in Chapter 6. An ontology-based documentation approach (introduced in Chapter 4) was used by participants to answer the remaining questions. For example, the participants that answered questions 1A, 1B, and 2 using file-based documentation would subsequently answer questions 3A, 3B, 4A, and 4B using ontology-based documentation.

In total 91 answers to the 7 questions were given by 26 participants when using file-based documentation. The researcher conducting the study read the 7 questions aloud to the participants. We asked all participants to search until they were satisfied with the time spent on an answer and its perceived correctness and completeness. Participants were instructed that this satisfaction should reflect their normal way of working.

We measured AK retrieval efficiency by recording how much *time* participants spent on accomplishing each task, namely, searching and providing an answer to

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

a question. Effectiveness was measured by recording the *recall* of participants, i.e. the completeness of their answers, and *precision*, i.e. the correctness of their answers. A complete answer (resulting in perfect recall) to questions 1A, 1B, 2, and 4B included multiple knowledge elements, e.g., two settings, three requirements, or four interfaces.

The ‘ground truth’ for evaluating recall and precision was verified in a pilot with two Océ professionals who did not participate in the study. They were asked whether an answer for a given question was complete and correct. We use "completeness" to refer to recall and "correctness" to refer to precision in the rest of the chapter for a better understanding.

The two professionals that participated in the pilot also proposed improvements to the question set. They evaluated whether each question was representative of the questions that software professionals at Océ normally ask and whether each question was relevant to software professionals in different roles. The questions were also evaluated on their representativeness and relevancy by five participants in a questionnaire after the experiment reported in Table 6.5 in Section 6.3. They evaluated all questions as relevant and representative for their jobs except for question 3A, which one participant evaluated as irrelevant and not representative.

The researcher conducting the study kept track of what participants indicated to be answers to a question. When a participant stopped searching, said s/he found an answer, or said s/he was satisfied, the researcher verified with the participant whether this was the final answer to the question.

We captured the search actions of participants by video recording their monitor screen. We used the think aloud method [115] and asked participants to think aloud when searching and recorded their voice in the video recordings.

2.2.2 Identification of Search Strategies and Prior Knowledge

We identified around 2,500 search actions in over 11 hours of video recordings. Table 2.2 details the different types of search actions that we identified and encoded from the videos. We collected the search actions used to find 90 of the 91 answers given by the participants. The video record of one participants answering one question was corrupted beyond repair and is thus excluded from our analysis.

Not all participants were talkative, so the think aloud recordings for some questions were more detailed than others. Also, some phrases and parts of sentences said in video recordings of 22 of the 90 answers could not be heard clearly due

2.2. DESIGN AND ANALYSIS OF SEARCH BEHAVIOUR STUDY

Table 2.2: Encoding of search actions from video recordings

Search action	Description and criteria for identification
Exploring directories	
Open Dir	Participant opens directory.
Inspect Dir	Participant has contents of directory on screen for 3 seconds or more.
Open Doc	Participant opens document.
Dir keyword search	Participant searches for keyword in the documents in a directory.
Inspect Dir search result	Participant inspects the list of documents found by using a keyword search in directory.
Exploring documents	
Scan section	Participant has content of document section or diagram on screen for 3-5 seconds.
Detailed scan	Participant has content of document section or diagram on screen for more than 5 seconds.
Scroll to section	Participant scrolls to a specific section and does not inspect intermediate sections.
Scroll to see section title	Participant scrolls to see the title of section currently being read.
View TOC	Participant looks at Table of Contents for 3 seconds or more.
Click TOC	Participant clicks on an entry in Table of Contents to navigate to section.
Keyword Search	Participant searches for keyword in document.
Inspect context of search result	Participant looks at keyword search result and surrounding text for more than 3 seconds.

to low sound recording volume and low volume of participants' voices. We however could often still infer what was said from the context of the search. One researcher spent 8 weeks to encode the search actions and transcribe think-aloud recordings from the videos.

From the identified search actions and think aloud verbalization we constructed Problem Behaviour Graphs (PBG) [73]. The construction of PBGs is a form of protocol analysis [35] which can be used to identify how people use their intelligence to solve problems in complex real-world environments [19]. In [19], Chen and Dhar used PBGs to model and investigate the cognitive process of people engaged in online document-based information retrieval. In our case the problem space consists of finding answers to questions using the document organisation, content, and the search functions of the documentation tools.

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

A PBG starts with the initial state of knowledge one has about a problem. In our case the initial state of knowledge was the question asked and the existing prior knowledge of participants about the documentation, its content, and the software system and project it specifies.

The initial knowledge state in a PBG changes to other knowledge states as the problem-solving process progresses. Problem-solving progressed when participants executed search actions in order to obtain new knowledge about the search problem. When a solution is found the problem-solving process ends. In our case the problem-solving ended when a participant answered a question.

Figure 2.1 shows one of the constructed PBGs in which a participant had to find settings for behaviour ‘*Alert Light*’ in order to answer question 1B. The time sequence of search actions is from top to bottom in this PBG. The knowledge states are represented by boxes with rounded corners, that each contain the additional knowledge acquired by the participant when searching for knowledge.

We identified four search strategies using PBGs which are detailed below with a concrete example. We could identify to which search strategy each of the 2,500 encoded search actions belonged. In most cases multiple search strategies were used when answering a question.

In the PBG example shown in Figure 2.1 the first search strategy used by the participant is to **explore the document organisation** in directory *SBD*. The documentation was organized by means of directories, documents, and sections. Part of the information in the contents of document sections was organised by lay-out and text notations, e.g., in the phrase “*REQ_1: users can save login credentials in Comp_3: UI*” which makes a requirement explicit. Information was organized in diagrams by means of UML notations that, e.g., denoted interfaces and interactions.

Figure 2.1 shows that using this strategy the participant finds document *SBD_Alert_Light* by inspecting the content of directory *SBD*. The title of *SBD_Alert_Light* relates to behaviour ‘*Alert Light*’ in question 1B, which indicates that this document contains information relevant for answering question 1B. The participant then opens *SBD_Alert_Light* and checks if it contains a reference to a dedicated settings document. In the next two search actions the participants scans for requirements and product information related to behaviour ‘*Alert Light*’ and one setting is found.

Exploring document organization is a search strategy that was used when searching for 88% of the answers in the study. Participants that spent time on exploring this document organisation would often quickly gather relevant clues about which locations contained answers to questions.

2.2. DESIGN AND ANALYSIS OF SEARCH BEHAVIOUR STUDY

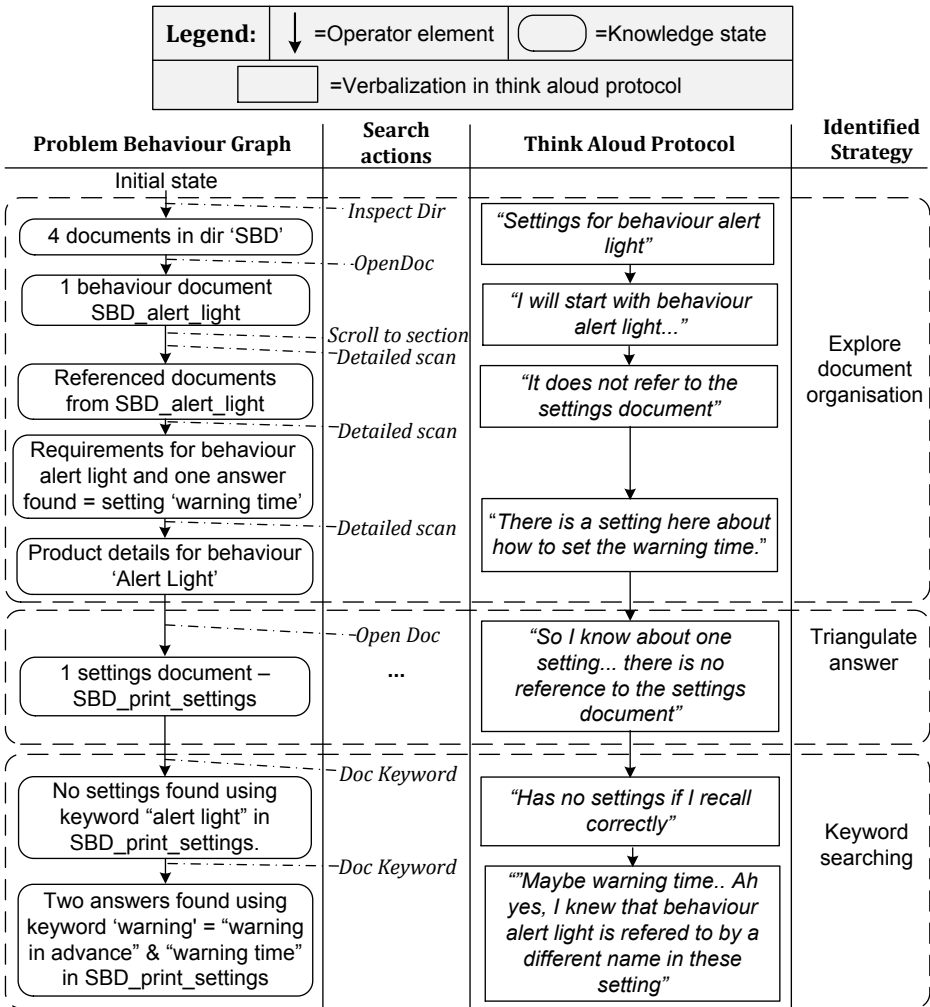


Figure 2.1: Problem Behaviour Graph of participant answering question 1B (Which settings have an impact on behaviour "Alert Light"?).

The organisation of documents does not always fully relate to the questions that document users have to answer. For example, none of the directory and document titles used in the study revealed where the decisions for question 3A and 3B could be found. Section titles inside two documents did explicitly relate to these

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

decisions, and the documents had to be opened to discover this. Participants could exhaustively explore the document organisation to discover documents and sections that related to a question, however, this took a lot of time for several participants.

Alternatively, participants would directly **search the expected locations of answers** by predicting in which location (directories, documents, and sections) they would most likely find answers to a question. Participants that used this alternative strategy directly navigated to certain locations at the start of a question, without exploring the available document organisation beforehand². Searching the expected location of answers is a search strategy that was used when searching for 29% of the answers.

The third and fifth sentences in the think aloud protocol in Figure 2.1 show that the participant thinks aloud about another document. After finding an answer in document *SBD_Alert_Light* the participant decides to open the other document *SBD_Print_Settings* so s/he can verify the correctness and completeness of the answers. We named this strategy '**triangulate answer**', as multiple sources are used to verify and improve the answer. This strategy was used by participants when searching for 11% of the answers.

The participant subsequently uses a strategy of **keyword searching** for the name of behaviour '*Alert Light*'. After an unsuccessful keyword search, the participant recalls from prior knowledge that the settings may not be mentioned by the exact name of behaviour '*Alert Light*', and starts to use a different keyword.

The file explorer, document editor, and UML tool used in the study provide keyword search functions that show their users which document titles, text fragments, and UML elements match a given keyword. We identified **keyword searching** as a strategy that participants used when searching for 62% of the answers.

²We observed from the search actions and think aloud statements that participants required 3 seconds or more to recognize and explore the document organisation when it was shown on their screen. After 3 seconds or more the participants acted upon the information by exploring the document organisation and they talked about this information, e.g. "*I see a settings document in this directory*". The participants did not actively explore the available document organisation if it was shown on their screen for less than 3 seconds. Instead the participants directly navigated to directories, documents, and sections in which they expected to find answers.

2.3 Using Prior Knowledge to Search under Uncertainty

In the think aloud recordings the participants voiced that they were uncertain about the correctness and completeness of 34 answers, out of the 90 answers (38%) given in the study. 13 of these 34 answers were actually correct and complete. Participants also voiced for 11 of the 34 answers that in everyday practice they would verify the answer with a colleague.

Typical remarks about this uncertainty are: "*It is difficult to know whether you found everything in the documentation. [I am] 70% sure of [my] answer*", "*Because searching was difficult I am not sure if this is [the] correct [answer]*", "*I think there is a 50% chance that I have found all answers*", and "*I have reasonable confidence that I have not missed [any parts of the answer]*".

We observed how participants used their prior knowledge to deal with their uncertainty. Prior knowledge was used to predict which documents might contain answers when the document organisation did not relate to the question. Participants were able to recall from prior knowledge what different spelling variations, synonyms, and acronyms existed for technical terms required in the search, and this enabled them to quickly find answers by keyword searching. Participants also used prior knowledge to recognize answers and to predict whether an answer was correct and complete.

Participants talked about how to use their prior knowledge when applying the search strategies identified in Section 2.2.2. For example, they voiced which documents might be relevant ("*I think only Sysref contains answers*"), which keywords to search for ("*I know that Alert Light is referred to by a different name*", also see Figure 2.1), and which answers were complete ("*This setting is the answer. I already knew this setting.*"). Participants acquired this prior knowledge by, e.g., having used the documentation, working on the software system, and by attending meetings, presentations, and conversations with other professionals.

In the next subsection we first describe how participants acquire prior knowledge in the study. In subsections 2.3.2 to 2.3.6 we report the different ways in which participants used their prior knowledge to search for answers. We also report cognitive biases that may occur during the use of prior knowledge.

We describe the gain when use of prior knowledge leads to complete and correct answers and the loss when it does not lead to answers. The gain is categorized as 'small' or 'large' in terms of time saved (compared to the average time spent on a question) and whether the use of prior knowledge helped participants to find complete and correct answers. The loss is similarly categorized as 'small' or

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

Table 2.3: Overview of prior knowledge, evaluation, and cognitive biases identified in study

Prior knowledge	Gain if correct	Loss if incorrect	Cognitive bias and possible underlying reasons
Answer is in location X	large	small	No bias identified.
Answer is not in location X	small	large	Availability bias: difficult to recall examples of answers found in unfamiliar location X .
Keyword X leads to answer	large	large	Availability bias: keywords that are often used for searching are familiar and more easily remembered.
Answer can be triangulated	large	small	No bias identified.
Answer is already known	small	large	Confirmation bias: focus on confirming known answer.

'large'. If a large gain means that participants found many answers in little time, then a comparatively large loss is that many answers were missed and much time was wasted.

We have summarized the findings in Table 2.3. The first column denotes what prior knowledge a searcher may have and the last column denotes what cognitive bias may occur when using this prior knowledge. Column '*Gain if correct*' denotes whether a small or large gain was observed when the prior knowledge was correct and led to answers. Column '*Loss if incorrect*' similarly denotes the loss when the prior knowledge was incorrect and did not lead to answers

2.3.1 Acquiring Prior Knowledge

Several participants voiced that they learn about how knowledge is organised in the documentation when searching. For example, one participant voiced: "*From the previous question I have gained knowledge about behaviour History*" and "*I already have seen that this knowledge is described in SBDs and not in SADs. So I already have an approach that works for [searching] requirements*". A participant explicitly voiced that this learning process was intentional: "*I would need to build up a kind of model in this environment, in documentation, to find an approach for searching. I need to open a few documents in order to come to that approach*".

We could observe that participants most often acquired knowledge about documentation by exploring the document organisation (one of the search strategies).

2.3. USING PRIOR KNOWLEDGE TO SEARCH UNDER UNCERTAINTY

Participants visited or ignored locations based on what they had learned from exploring the document organisation during preceding questions. Participants also used keywords that were successful in earlier searches and used keyword spelling variations they found when exploring document organisation.

2.3.2 Predicting Which Locations Contain Answers

Several participants voiced in which locations they expected to find answers. For example, four participants voiced in which locations they expected an answer to the first question 1A before they started to search: "*I will first look in SBD*", "*I will look in SBD_history, it has standards settings*", "*Behaviour is in SBDs*", and "*Then I would look in the requirements*".

After these statements the participants directly navigated to directories, documents, and sections instead of exploring the available document organisation. They acted on their prior knowledge about the documentation. One participant explicitly voiced this: "*From my knowledge I know I should look in SBD_history and SBD_print_settings. I would not expect something in SBD_docbox ... I however do not claim that this is indeed the case*". Such experience provides a starting point for the search.

The participants intuitively predicted from prior knowledge that certain locations contained relevant information or an answer because they found (similar) information or answers there before. This is an availability heuristic, described by Tversky and Kahneman in [108], which people use to estimate the probability or frequency of an event by recalling occurrences of similar events from their memory.

Correctly predicting that a location contains an answer resulted in a large gain. Namely, participants that directly navigated to locations found 19 answers to questions in the expected location and spent, on average, 37% less time compared to the average time spent on searching these answers.

Incorrectly predicting that a location contained an answer resulted in a small loss compared to the gain above. Namely, one participant wasted 70 seconds searching for an answer that was not in the expected location. The participant however still spent less time than average to answer this question. After the unsuccessful search in the expected location, the participant used other search strategies (explore document organization and keyword searching) and then found the answer. He used an agile search approach by switching to a different search strategy after the initial search strategy did not work.

2.3.3 Predicting Which Locations do not Contain Answers

Prior knowledge was also used to predict that an answer could *not* be found in certain locations, namely, in specific directories and documents. Participants ignored locations, i.e., they did not search in locations where it was unlikely to find an answer. This helped to cut down the search space and thereby save time. However, participants also gave incomplete and incorrect answers to questions because they ignored certain locations.

One of the participants said that a document containing answers to questions 1A and 1B was not related to these questions. During question 1B he voiced: "*SBD_print_settings has nothing to do with [behaviour] 'alert light'. This I know.*". Three participants ignored locations with answers and gave no explicit reason as to why they ignored the locations. Another participant said that he decided to not open the Sysref document because he was not very familiar with it: "*I cannot do much with the Sysref ... I do not really know the Sysref that well.*".

In [108] Tversky and Kahneman describe how estimating the occurrence of an event is affected by the ease with which one can bring instances of this event to mind from personal experience. Events that are familiar to a person are more easily retrieved from memory than less familiar events, and this biases the use of availability heuristics. The participant that explicitly voiced that he was not familiar with a location had difficulty in recalling examples of answers in this location. The participant chose to visit locations he was more familiar with and not the location that he was less familiar with. This suggests that the participant missed answers because of availability bias.

Correctly predicting that a location can be ignored, i.e. ignoring a location that indeed does not contain answers, resulted in a small gain, namely, time was saved by not having to inspect this location.

Incorrectly predicting that a location can be ignored however resulted in a large loss compared to the gain above. Namely, participants did not find complete answers to 7 questions because they ignored the location containing the answer. Moreover, they wasted time searching for answers in other locations that they did not ignore. When these participants did not find answers they did not reconsider and check the locations they ignored. We asked all participants in the study to search until they were satisfied with the time spent and answer found. In this case the participants decided to stop searching without finding an answer within reasonable time.

2.3. USING PRIOR KNOWLEDGE TO SEARCH UNDER UNCERTAINTY

2.3.4 Predicting Which Keywords Lead to Answers

The names of certain knowledge elements, e.g. decisions, settings, and sub-systems, can be recorded using different spelling variations and acronyms. For example, the component in questions 2 and 3A has three spelling variations in the documents; ‘*settings editor*’, ‘*settingseditor*’, and ‘*setting editor*’, and one acronym; ‘*SE*’. Keyword searching for only one of these spelling variations does not return all locations that mention this component.

A participant voiced this problem quite clearly after keyword searching for requirements realized by component settings editor (question 2): "*I am not sure if [my answer] is complete. There could be requirements that do not contain the name ‘settings editor’. Or [the name] is recorded differently*". Another participant voiced concerns about how to spell the interface for question 4A: "*JJ-I. I wonder if there are different ways of writing it*". One participant emphasized the importance of prior knowledge in this situation: "*So context, about how we call certain things within Océ, is really needed to search fast*".

Moreover, certain keywords only led to part of the answers, because these keywords were not recorded in all descriptions of these answers. This was often the case for keywords that indicated a type of knowledge. For example, only part of the decisions could be found using keyword ‘decision’ because several descriptions of decisions did not contain the actual word ‘decision’. People used prior knowledge to predict the ‘coverage’ or ‘frequency’ of keywords.

We observed that 8 of the 26 participants used part of a name in their keyword search, which allows multiple spelling variations to be covered in one keyword. For example, they used keyword ‘*editor*’ to search for component ‘*settings editor*’ in question 2 and 3A. One participant voiced this use of partial keywords for question 2: "*Maybe I can search for something like ‘setting’ or ‘editor’*". Participants that used partial keywords however found much knowledge that was irrelevant for their question, and this resulted in lower average efficiency than the use of full names when keyword searching.

The participants had a clear preference for using certain keywords over others. They used keyword searching when trying to find 56 of the 90 answers, yet in only 7 cases they searched for all the keywords phrased in the question they tried to answer. Participants voiced that they are familiar using certain keywords: "*I do not know how this is always written in the text so I always search for settings editor concatenated and settings editor with a space in between*" and "*Normally I would have to search for keyword ‘decision’*". One participant used keyword ‘*requirement*’ to answer questions 1B and 2 because he had successfully used the keyword during preceding question 1A: "*I already have an approach that works*".

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

for *[searching] requirements*".

Several participants in our study voiced that they used certain keywords because they were familiar with using these. They recalled from experience that these keywords could be used to find answers, however, these keywords did not always lead to answers in this study. This selection of keywords based on familiarity suggests an availability bias.

Accurately searching for keywords, i.e. using keywords that lead to descriptions of answers, resulted in a large gain. Namely participants found 34 answers to questions by keyword searching even though they spent 17.7% more time than the average for these questions.

Keyword searching allowed participants to find answers without having to spend a lot of time on exhaustively exploring document content. Several participant voiced this as their motivation: "*I find a lot of text here so I will switch to keyword searching*", "*It is a relatively huge document so what I will do is a keyword search*", and "*I have no other option than to use keyword searching*".

Inaccurately searching for keywords, i.e., using keywords that do not lead to descriptions of answers, resulted in a large loss compared to the gain above. Participants could not find 22 answers to questions via keyword searching and spent 8% more time on these questions than the average time required for these questions. These participants also used other search strategies, however, they gave up searching without finding a complete answer to 15 of these 22 questions. They gave up partially because relatively much time was spent on keyword searching without finding an answer.

2.3.5 Predicting Whether Answers can be Found in Multiple Locations (Triangulation)

After finding an answer in some location, several participants tried to verify whether the answer was complete and correct by searching in other locations. This strategy is called *triangulation*, which we describe in Section 2.2.2. This strategy is applied when, after a participant finds an answer in one location, s/he navigates to other locations to verify and improve the answer.

Participants predicted from prior knowledge whether they could find the same - possibly more complete - answer in another location. For example, one participant first found an answer to question 4B by searching in document Sysref and then voiced "*there should be a diagram here somewhere*". The participant then visited the UML document to find a more complete answer.

2.3. USING PRIOR KNOWLEDGE TO SEARCH UNDER UNCERTAINTY

The answers for questions 1A, 1B, 4A, and 4B were described in multiple locations. For example, participants looking for settings whilst answering question 1B often could not find all the settings in *SBD_Alert_Light*. Settings were not very explicit in this SBD and the section titles did not clearly show where an answer could be found. Participants who also looked for an answer in *SBD_Print_Settings* would however find the settings more explicitly recorded.

Accurately triangulating answers, i.e. an answer is improved by searching for the same answer in another location, resulted in a large gain. Participants improved the completeness and correctness of 3 answers by triangulating the answers, even though they spent 95% more time on average (one participant spent 420% of the average time finding 1 of the 3 answers). 16 out of the 90 answers (18%) in this study could have been more complete and correct if the participants had triangulated their answers.

Inaccurately triangulating answers, i.e. searching for the answers in another location but not finding relevant information to improve the answers, resulted in a small loss compared to the gain above. Six participants triangulated one of their answers in another location but did not improve the answer. They spent on average 23.6% more time than other participants that answered the same questions.

2.3.6 Estimating Whether an Answer is Complete and Correct

Most participants had extensive experience using the documents and building the software specified in it. Several of these participants had a good idea what would be the likely answers to the questions. When participants said that they knew an answer from prior knowledge, we instructed them to nevertheless answer the question using the documentation.

Participants used their prior knowledge about possible answers to recognize answers while searching and to estimate the completeness and correctness of the answers they found. This often worked well, however, in some cases the prior knowledge about possible answers was incomplete. We found that 5 out of the 26 participants made a false assumption because of this.

These 5 participants falsely assumed that an answer they knew from prior knowledge was complete and correct, whilst in reality their answer was incomplete. Two of these five participants gave an answer that was both incomplete and incorrect.

For example, one participant voiced: *"I think my answer is right and I am thus satisfied. I however already knew this was the answer"*. Another participant who

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

had to find two settings for question 1B voiced: "*this is indeed the only setting*". All these participants had the missing parts of their answer on screen for some time but ignored this.

The search behaviour of these 5 participants was affected by confirmation bias [74]. The participants searched for answers that they knew from prior knowledge and confirmed that these answers were recorded in documentation, i.e., they confirmed their prior beliefs. In the process they ignored other answers and information that was inconsistent with these beliefs.

Accurately estimating whether an answer is complete and correct based on prior knowledge resulted in a small gain. 10 participants gave a correct and complete answer that they claimed to already know from prior knowledge. These participants spent, on average, 3% less time to find this answer than other participants.

Inaccurately estimating that an answer is complete and correct based on prior knowledge or prior belief, resulted in a large loss compared to the gain above. Five answers to questions that were assumed to be correct and complete by 5 participants were in fact incomplete and incorrect. The participants answered the five questions in 74% of the average time that other participants spent on these questions. This was because they searched briefly or they stopped searching immediately after finding an answer known from prior knowledge.

2.4 Lessons Learnt

2.4.1 Lessons for Documentation Users

Prior knowledge can be used to quickly find correct answers to questions when the document organisation does not support the questions. We found that the use of certain types of prior knowledge yields larger gains than the use of other prior knowledge. Moreover, there is a difference in losses when incorrect predictions are made based on prior knowledge.

Table 2.3 shows that it was rewarding for participants to visit the expected location of an answer and to triangulate an answer in multiple locations known from prior knowledge. If this prior knowledge proves to be incorrect the loss is small, and it is therefore relatively safe to use. Using prior knowledge to predict which keywords lead to an answer often yields a large gain, however, incorrect predictions may result in a large loss and this prior knowledge should thus be used with caution. Ignoring certain locations or estimating that an answer is

complete and correct from prior knowledge yields a small gain and such cognitive biases increase the chance of a large loss.

Being more aware of cognitive biases can prevent the aforementioned losses. It may be hard to remember examples of answers being recorded in a certain location because one is not familiar with this location, however, this does not imply that the location indeed contains no answers. Certain keywords are easily remembered because they are often used and familiar. Using these keywords in searching may however be counter-productive. Prior knowledge about the answer to a question may be incorrect, incomplete, or outdated.

Existing prior knowledge can be evaluated and updated, by thoroughly exploring document organisation and content, i.e., conducting empirical investigation and seeking disconfirmatory evidence [91]. This prevents inaccurate predictions and cognitive biases later on. An additional benefit of exploring the document organisation and content is that it can remove search uncertainty and the need for using prior knowledge. A searcher may find document organisation that is fitting for the question asked, and this organisation often leads to complete and correct answers.

2.4.2 Lessons for Documentation Writers

We observed several causes for search uncertainty:

- Document organisation does not relate to a question because document writers do not plan the document organisation to answer all questions that could arise.
- Documents that are not searched might contain answers.
- The same knowledge might be referred to by multiple spelling variations and acronyms.
- The type of knowledge might not be consistently recorded.
- Complementary knowledge might be described in multiple locations.

Consequently, a searcher might only become certain that an answer is correct and complete when all text in the available documentation is read. We observed that none of the participants exhaustively inspected all available document contents. Participants either quickly found answers using document organisation that was fitting for the questions, or they used their prior knowledge to predict which locations and keywords were relevant when searching.

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

Searchers make predictions from their prior knowledge to deal with search uncertainty. These predictions can however be inaccurate and are prone to cognitive biases. This results in inefficient and ineffective use of documentation, and in turn lowers the incentive to spend resources on producing good documentation, creating a vicious cycle [77]. Moreover, incomplete and incorrect answers are used to build software and may result in costly errors.

Addressing causes for search uncertainty removes the need for searchers to use prior knowledge and lowers the chance that cognitive biases occur. Creating a document organisation that fully relates to commonly asked questions removes much search uncertainty. Introducing spelling conventions and consistently recording what type of knowledge is described makes keyword searching more efficient and less error-prone. Recording the same type of knowledge in one location prevents scattered descriptions of the same knowledge that can become inconsistent over time, and in turn removes the need for searchers to triangulate answers.

These solutions are difficult to realize when writing and maintaining documentation in a linear file-based format with multiple authors. As an alternative, ontology-based documentation may address above issues, whilst it also provides benefits from the use of explicit semantics and non-linear organization of knowledge. We explore this alternative in the remainder of this thesis.

2.5 Threats to Validity

The participants also used ontology-based documentation to answer questions as part of the experiment reported in Chapter 6. We did not find evidence that the use of the ontology-based documentation approach had an influence on how participants used the reported search strategies and their prior knowledge in file-based documentation.

To verify the consistency of the used encoding scheme, described in Section 2.2.2, two researchers independently applied the scheme to identify search actions from the videos. After applying the scheme they checked if they came up with the same set of search actions.

We constructed PBGs using the format depicted in [19] and Figure 2.1. We observed that participants acquired new knowledge with each executed search action. For example, one participant voiced: “*I have already seen that this knowledge is described in SBDs and not in SADs.*”. As such the participants did not return to a previous state of knowledge, which is proposed as a PBG modelling construct by Newell and Simon in [73] but not applied in our study.

The use of SADs, SBDs, Sysref, and design documents, and the search functions of the tools used for searching these documents, can be considered generic documentation practice in industry. This suggests that the identified search strategies are also generalizable to other software projects.

The evaluation in terms of efficiency and effectiveness in this chapter is specific to the questions and Océ documentation used in this study. The use of prior knowledge and cognitive biases that may occur are however largely independent of the question asked and documentation searched. The use of prior knowledge and impact of cognitive biases also apply when searching in software documentation in other domains and is therefore generalizable.

2.6 Related Work

How knowledge is used in software documentation is systematically reviewed in [33], and in this section we only discuss related work on using prior knowledge to search software documentation.

Chen and Dhar describe in [19] how prior knowledge is used during online document-based information retrieval and how prior knowledge affects the selection of search strategies. In [86] Shute and Smith identify the use of prior knowledge as 'subject-dependent expertise' for searching bibliographic databases. For example, the 'known-item-instantiation' strategy described in [19] uses subject-dependent expertise. Shute and Smith describe how participants talk about using their intuition and "gut feeling", which suggests that they use their prior knowledge.

In [24] de Boer and van Vliet describe how software professionals in the same team have similar mental models of documentation. They have a shared understanding of the contents of these documents. Moreover, the development process affects the level of shared understanding within a team. Such mental models of documentation are part of the prior knowledge that we discuss in this chapter.

In [65] LaToza *et al.* describe how developers rely on implicit code knowledge and spend much effort to maintain a mental model of code. Developers recall details, e.g., about the architecture and design of their code, as part of their prior knowledge, i.e., they know code details by heart. Moreover, they often do not consult documentation to check if their mental model is consistent with documentation.

In [100] Tang describes how software design is affected by cognitive biases. Designers use prior beliefs and intuition to make judgements, and cognitive biases

CHAPTER 2. SEARCHING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

may occur because of this. In [17] Calikli *et al.* investigate how confirmation biases during software testing can be prevented.

In [91] Stacy and Macmillian describe how software professionals develop and use their mental models during software engineering activities and how this is influenced by cognitive biases. For example, code features that are easily remembered by software professionals may be judged to occur more frequently than other features due to availability bias. They suggest that keywords that are very long, occur in recently read documents, or occur in code recently worked on may be more easily remembered than other keywords, and this may cause availability bias. Similarly, we observed that software professional tend to search keywords that are familiar and easily remembered from prior knowledge, and that this may result in availability bias.

In [58] Korkala and Maurer identify communication waste, e.g., outdated and scattered information, in a software project. The software documentation in our study is used to communicate knowledge. As such, the losses during the use of prior knowledge, described in sections 2.3.2 through 2.3.6, can be regarded as communication waste. The identified causes for search uncertainty can in turn be regarded as causes for communication waste.

In [96] Su *et al.* used Information Foraging theory to explain how software professionals search for architectural knowledge in document sections using several foraging styles. Information Foraging theory tries to explain the search behaviour of people in terms of cost and reward when navigating an information topology. The study in [96] however does not evaluate whether the observed search behaviour is time-efficient and does not focus on use of prior knowledge.

In [56] Ko *et al.* report an exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. Their analysis of information seeking behaviour of software developers relates to our work in which software professionals also exhibit information seeking behaviour. However, software maintenance information is searched in source files and on the Internet, whereas in our work software professionals search for AK in SA documents.

2.7 Conclusions

File-based SA documents are used to capture and communicate AK in software projects. It is important that this AK can be retrieved efficiently and effectively, to prevent wasted time and errors that negatively affect the quality of software. The organisation of SA documentation typically does not support all of the questions asked by software professionals. This introduces search uncertainty and

makes it hard for software professionals to find complete and correct answers within reasonable time.

We conducted an industry study to investigate how software professionals search for AK in SA documentation. We found that professionals use their prior knowledge to find answers when the document organisation did not relate to the questions they had to answer. Prior knowledge was used to make predictions about the location of AK, which keywords can be used to search relevant AK, and whether the AK found is correct and complete.

Using prior knowledge is often time-effective, however, inaccurate predictions and cognitive biases can lead to inefficient and ineffective AK retrieval. Availability bias may cause searchers to ignore locations and keywords that they are not familiar with, even though these locations and keywords may lead to answers. Using prior knowledge is also prone to confirmation bias when searchers mainly focus on confirming the answers that they already know from their prior knowledge.

Awareness of these cognitive biases may reduce the likelihood that they occur when searching for AK in SA documentation. Searchers can evaluate and update their existing prior knowledge by spending time on exploring the document organisation and content, which further reduces the probability that cognitive biases occur. Addressing causes for search uncertainty when writing SA documentation removes the need for searchers to use prior knowledge and in turn prevents that cognitive biases occur.

3

Organising and Retrieving Architectural Knowledge in File-based Documentation

In this chapter we review literature to find out how AK is typically retrieved from file-based documentation (RQ1). We found that file-based documents have a linear organisation of AK whilst document users do not necessarily retrieve AK following the same organisation. Users may not easily find AK that is outside the document organisation, however, creating a document organisation that supports the AK retrieval needs of all users is difficult and introduces redundant and scattered AK descriptions. AK retrieval challenges reported in literature stem from limitations of the linear file-based document organisation.

The previous chapter showed that industry professionals experience AK retrieval challenges when the document organisation does not support their questions. In Section 3.1 of this chapter we study literature to find out how SA documents are typically organised, why they do not support the questions of all AK users, and to identify AK retrieval challenges reported in literature. We shortly discuss in Section 3.2 to which extent hypertext documentation can alleviate the identified AK retrieval challenges. The next chapter details how the challenges may be alleviated by ontology-based documentation.

3.1 File-Based Documentation and its Issues

In their highly influential paper on multi-dimensional software decomposition [105], Tarr *et al.* describe how traditional formalisms in software engineering can only provide a single “dominant” dimension when achieving separation of concerns. Use of a single dominant dimension, by software decomposition based on

CHAPTER 3. ORGANISING AND RETRIEVING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

e.g., quality, functions, development tasks, or data objects, causes problems with reuse, traceability, comprehension, evolution, and maintenance. These problems not only apply to the software itself, but also to its documentation.

Parnas and Clements argue [78] that documents should be designed and structured with separation of concerns in mind; each aspect of a system is described in one section. File-based documentation can achieve this separation by using, for example, a view-based structure [9, 21, 2].

Each view provides a ‘cross-section’ of AK. Views are useful to stakeholders who are interested in different cross-sections of AK. Cross-referencing of AK between views can help to make interrelated AK traceable and retrievable.

The separation of concerns achieved through a particular set of architecture views makes the retrieval of certain knowledge – knowledge contained in one view – relatively easy, but at the same time it makes the retrieval of knowledge scattered across views difficult. This is a wicked problem: choosing a different set of views does not solve the issue, but simply moves it elsewhere. This problem is recognized by Rozanski and Woods in [81], where the notion of perspectives is introduced next to that of views. Perspectives serve to organise specific types of knowledge across views.

As the number of different stakeholders and their unique needs for AK increase in large and complex projects, there is also an increase in misalignment between the AK needed by stakeholders and how they may retrieve this AK from file-based documentation. In practice, most stakeholder concerns are addressed by a small documentation subset that is different for each concern [57]. Moreover, existing approaches for documenting decisions only frame part of the stakeholders concerns related to decisions [112]. Extensive use of cross-references between scattered AK or (alternatively) redundant recording of AK in file-based documents makes AK retrieval and maintenance impractical and error-prone.

The questions about AK that documentation users may ask based on their concerns are illustrated in the right-hand side of Figure 3.1. The questions are about certain types of AK, e.g., decisions and requirements, and relationships between AK, e.g., ‘impacts’ and ‘realized by’. Relationships between AK show how an architectural element is connected to or associated with the rest of the architectural design. For example, a developer may want to find all requirements realized by a component s/he has to build, whilst an architect may want to find all decisions that impact the same component during impact analysis.

The left-hand side of Figure 3.1 illustrates how a linear organisation of AK in a table of contents supports file-based document users in finding a limited set of relationships between AK. As a result, only one out of three questions asked by

3.1. FILE-BASED DOCUMENTATION AND ITS ISSUES

document organisation	supported AK relationship	match?	required AK relationship	questions of document users
Table of Contents 1. Functional requirements 1.1 Requirement 1 - <i>Login</i>	Requirement → subsystem			
1.1.1 Subsystem <i>FrontEnd</i> [...] 1.1.1.5 Decision D5	subsystem → decisions	No	decisions → design alternatives decisions → (related) decisions	"I need to find all alternatives and decisions that are related to decision D5"
2. Performance 2.5 Decisions	quality attribute → decisions	Yes	quality attribute → decisions	"I need to find all decisions that impact performance"
3. Architecture design 3.5 Subsystem <i>FrontEnd</i> 3.5.2 Component <i>GUI</i>	subsystem → component	No	component → requirements	"I need to find all requirements that are realized by component GUI"
3.8 Maintenance 3.8.3 decision [...]	quality attribute → decisions			

Figure 3.1: Mismatch between supported and required AK relationships in file-based document organisation

the document users is supported in this AK organisation.

For instance, the organisation does not detail where every type of AK can be found, e.g., design alternatives. Moreover, the relationship between components and requirements, necessary to answer the question about component *GUI*, is missing in this organisation. Extending the AK organisation to support this question would introduce redundant and scattered descriptions of either requirements or components.

Indexing additional relationships (or 'cross-sections') between AK in a file-based document organisation introduces redundant and scattered AK descriptions. Relationships between AK that are not indexed by the document organisation have to be searched inside document contents. It is however difficult to make document content unambiguous [77] and organise the AK therein such that it is successfully communicated to users with different backgrounds [83].

Explicitly describing relationships between AK makes the AK traceable. Empirical evidence is given by Shahin *et al.* in [85] and Javed and Zdun in [53] that improved traceability leads to better architectural understanding. Lack of traceability in SA documentation is considered a major problem in industry practice [80].

CHAPTER 3. ORGANISING AND RETRIEVING ARCHITECTURAL KNOWLEDGE IN FILE-BASED DOCUMENTATION

In [51] Jansen *et al.* identify AK retrieval challenges that partially stem from above issues with organising AK. We describe in Section 4.2 how the challenges can be alleviated by the ontology-based approach.

1. *Architecture documentation understanding*
Document understandability becomes more challenging when documentation size increases in large and complex systems [21]. The original intention of the authors is often lost.
2. *Locating relevant architectural knowledge*
Knowledge is often spread over multiple documents [23] which makes it hard to locate AK, especially if documents lack finer details.
3. *Support for traceability between different entities*
Providing traceability between documentation sources is difficult [49]. Text and tables are limited in communicating different relationships.
4. *Support for change impact analysis*
Because decisions, requirements, and their relationships are usually not explicit, it is often very hard to reliably analyze and predict the impact of changes to the architecture.
5. *Assessment of design maturity*
Architecture design is difficult to evaluate if there is no status overview of the conceptual integrity, correctness, completeness, and buildability of the architecture [111, 9].
6. *Credibility of information*
AK often changes in large and complex systems and the cost to update is sometimes prohibitive [51]. Documentation is quickly outdated and its users lose confidence in its credibility [66].

Problems related to the above issues and challenges are reported by Rost *et al.* in a recent survey [80] on SA documentation among practitioners working in 33 companies around the world. The top three reported problems with the representation of AK in the documentation that 109 of these practitioners work with are 1) *inconsistent and missing structure*, 2) *scattered information*, and 3) *missing traceability*.

3.2 Hypertext Documentation and Its Issues

Hypertext and wiki systems have been used in some software organisations for SA documentation. The use of tags and categories can help to organise knowl-

edge. However tags quickly lose meaning when used arbitrarily. Hypertext is also known as nonlinear text [22], yet its organisation remains linear with the use of categories.

Hyperlinks provide cross-referencing by pointing to information, however, the pointers do not specify the meaning of relationships. Without explicit semantics, not all AK user will be able to understand an organisation of AK by means of hyperlinks.

Several researchers [32, 22, 39] report that users of hypertext documents feel 'lost' and have difficulty gaining an overview of the material being read and how this material is interrelated [106]. Likewise, users of wiki-systems may also experience a lack of structure when navigating and finding relevant information [14].

Hypertext systems address AK retrieval challenges 2 and 3 (in Section 3.1) to some extent using hyperlinks and challenge 6 using version control, e.g., in wikis. However, because hyperlinks do not have specific semantics, they are not practical for filtering and querying AK based on the properties of relationships between AK. This is necessary for effectively addressing AK retrieval challenges 1, 4, and 5.

Semantics can be conveyed by named hyperlinks in hypermedia systems [118], hyperlinks in knowledge-based hypertext [70], and labelled links in spatial hypertext systems. Solis *et al.* describe a spatial hypertext systems for AK retrieval in [90] and its qualitative evaluation in [89], which is the only study on using spatial hypertext for AK retrieval that we know of.

3.3 Conclusion

We studied literature to identify how AK is typically retrieved from file-based documentation. Many AK retrieval challenges reported in practice stem from limitations of linear file-based documentation organisation. This AK organisation makes it hard to separate concerns, and comprehensively organise interrelated AK to support the AK needs of all document users. Hypertext-based documentation used in practice lacks the semantics to fully address the limitations and challenges of file-based documentation. This suggests that there is room for improving AK retrieval practice.

4

Ontology-based Architecture Documentation Approach

In this chapter we investigate how an ontology can be used for retrieving AK from SA documentation (RQ2). We first give background information on the use of ontologies for organising and retrieving AK. We then introduce an ontology-based documentation approach that consists of a software ontology and semantic wiki tool that we optimized for SA documentation.

Section 4.1 details on ontologies for SA documentation in related work and how an ontology can be used to organise and retrieve AK as part of an ontology-based SA documentation approach that we propose. The proposed approach makes use of a semantic wiki, and we describe in Section 4.2 how it was adapted for storage and retrieval of SA documentation, and how it can address AK retrieval challenges. Section 4.3 describes in detail how AK in SA documentation content is annotated in the semantic wiki. Section 4.4 discusses related work and Section 4.5 concludes this chapter.

4.1 Software Architecture Ontologies

“An ontology” explicitly specifies the conceptualization of a domain [41], i.e. “an ontology” refers to a formal domain model in which concepts and relationships among concepts are described [69]. Ontologies enable a hierarchical classification of interrelated domain concepts and can be represented using an Resource Description Framework (RDF) Schema or the more expressive Web Ontology Language (OWL). The use of RDF makes ontologies human readable and machine-interpretable, allowing querying of and inference over knowledge.

CHAPTER 4. ONTOLOGY-BASED ARCHITECTURE DOCUMENTATION APPROACH

Ontologies, RDF, and OWL are part of the semantic web paradigm which aims to support more advanced knowledge management systems in which knowledge is retrieved via query answering (replacing keyword-based search) and presented in a human-friendly way [5]. Several ontologies and domain models have been proposed in recent years for expressing AK in order to capture, manage, and share architectural design decisions explicitly [84] as well as providing a common vocabulary and a level of precision needed for making architecture decisions [4, 60] and reusing architecture documents [119].

In this study we use the lightweight software ontology from [104] for annotating knowledge in architecture documents. We chose to use the lightweight software ontology because it is a general-purpose ontology; it contains architectural concepts that are commonly documented in a software project [104]. This ontology was built to support use cases around typical activities of architects [103]. The lightweight ontology is designed to be flexible so that it can be adapted for specific application domains. More information about the core elements of such an ontology can be found in [103].

Various other general-purpose ontologies have been proposed in [8, 119, 63, 4, 68] for describing commonly documented AK concepts in software projects. Many AK concepts in the lightweight software ontology are also described in other general-purpose ontologies, e.g., requirements in [8], architecture element such as components, subsystems, and interfaces in [119] and [63], and all aforementioned AK concepts together with decisions in [4] and [68].

Figure 4.1 depicts the classes and relationships in the lightweight software ontology¹. Classes that were added to support the AK concepts used in one of the experiment domains are appended with “(Océ)” and are explained in Section 6.1.3. We added concepts ‘Wikipage’ and ‘Diagram’ to support storage of SA documentation. We illustrate the full ontology in Figure 4.1 with a software development scenario below (the classes are marked boldface and the relationships are marked italic):

A software architect makes a **decision** that **non-functional requirement** ‘configurability’ is *realized by* the **architecture**. The **decision** *results in* **behaviour** ‘user preferences’ which *satisfies* the **non-functional requirement** ‘configurability’ and a new **functional requirement** ‘set user preference’. When a software engineer implements **behaviour** ‘user preferences’, s/he needs to know which **settings** can be *changed by* and *stored by* this **behaviour**. S/he also needs to know the **interfaces** that are necessary to *realize* the **behaviour** and possibly the details on the **components** or **subsystems** that *offer* these **interfaces**. When implemented, the **behaviour** can be tested using the **requirements** that are *re-*

¹See <http://www.archimind.nl/oce-ontology.owl.xml> for OWL source file of this ontology.

4.1. SOFTWARE ARCHITECTURE ONTOLOGIES

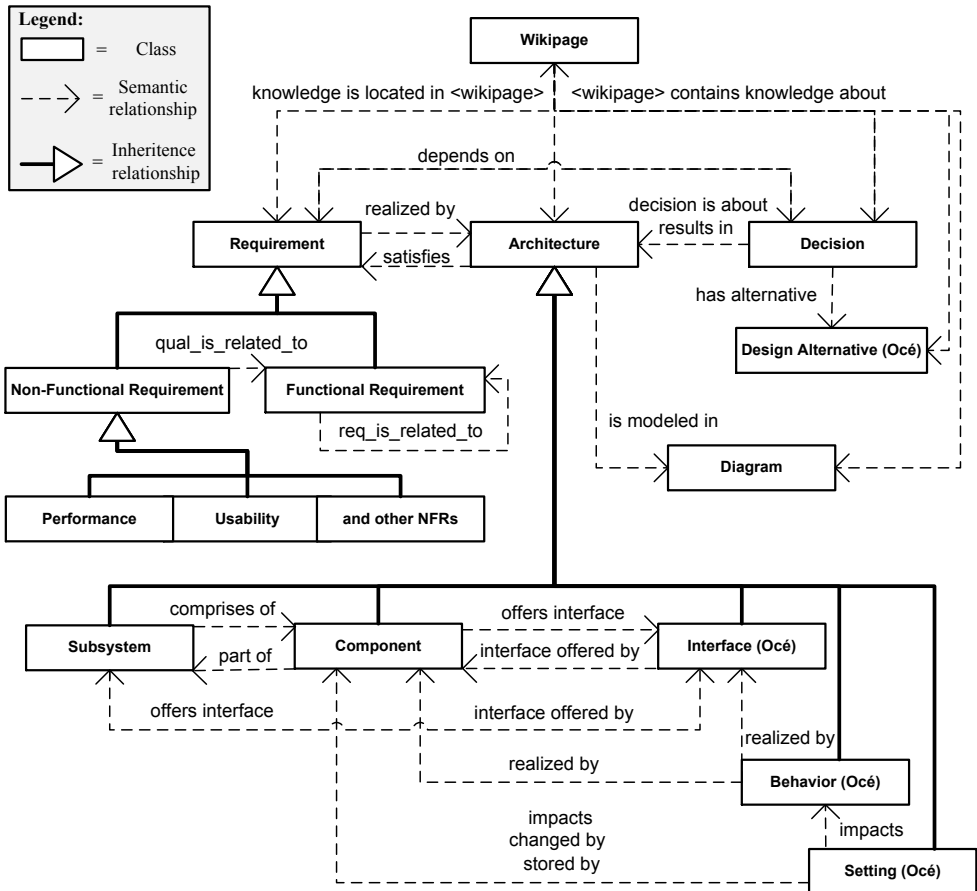


Figure 4.1: Software ontology adapted for the Océ experiment domain

alized by the **behaviour** and under various **settings** that *impact* it. **Wikipages** that *contain knowledge about* the aforementioned AK can provide additional context.

The relationships and classes in the ontology are used for organising AK. Relationships between classes support documentation users in finding relationships between AK. Each distinct ontology class and relationship has properties and descriptions that explicitly define their meaning, allowing different AK users to interpret them consistently and unambiguously. In the rest of this chapter, and most of the thesis, we refer to relationships in the ontology as '*semantic relationships*', because the names and properties of these relationships clearly convey

CHAPTER 4. ONTOLOGY-BASED ARCHITECTURE DOCUMENTATION APPROACH

their meaning (as opposed to hyperlinks, see Section 3.2), and to make clear that we do not write about another type of relationship.

4.2 ArchiMind Semantic Wiki

The use of a semantic wiki allows for navigation and presentation of classes and semantic relationships in an ontology. A semantic wiki can provide benefits similar to traditional wiki-like systems for documentation, such as centralised storage and access, text editing features, versioning, and collaboration mechanisms.

We used the OntoWiki tool [6] as the basis for the tool in our ontology-based approach. OntoWiki is open source and aims to support collaborative knowledge engineering. OntoWiki is similar to existing wiki systems (e.g., Wikipedia) and additionally it offers web-based visualization and management of (ontology and its instances in) knowledge bases and semantic-enhanced search facilities.

We based our choice for OntoWiki on evaluation of semantic wikis by Hoenderboom and Liang [48] and Tamburri [99]. In [99] Tamburri used a literature study to identify which requirements a semantic wiki for software knowledge management should satisfy. OntoWiki satisfied most of these requirements compared to other semantic wikis at the time, most notably faceted browsing, different views, ontology browsing, semantic inference, and requirements for social collaboration [99]. Hoenderboom and Liang show in [48] that OntoWiki provides many useful features for requirements engineering, especially semantic search and text annotation features.

Adaptations were made to version 0.9.5 of OntoWiki in order to optimize it for storage and retrieval of architecture documentation. We named the adapted version ‘ArchiMind’. See <http://www.archimind.nl/archimind/>² for a demo of ArchiMind.

Figure 4.2 depicts part of the ArchiMind GUI in which red labels highlight the different UI parts. Label A highlights the class navigation panel, used to retrieve all instances of an ontology class. The class navigation shows the classes of the ontology in Figure 4.1. The subclasses of Architecture and Requirement are not shown in the panel. These subclasses (denoted by their inheritance relationships to superclasses Architecture and Requirement in Figure 4.1) can be expanded by clicking on the arrowhead to the right of the name of the superclasses. Label C highlights a list with instances of class Requirement that were retrieved using the class navigation panel.

²The ontology shown in the demo is different from the experiment ontology.

4.2. ARCHIMIND SEMANTIC WIKI







A	C	D	
Navigation: Classes Search in Navigation Architecture Decision Design alternative Diagram Requirement Wikipage	<ol style="list-style-type: none"> <li data-bbox="423 285 974 342">  Availability Non-Functional Requirement <li data-bbox="423 352 974 409">  Choose representation Functional requirement, <li data-bbox="423 418 974 795">  Compatibility Non-Functional Requirement, knowledge is located in  24 Appendix - Push and Pull data Wikipage content Push data: The first solution we had considered was pushing data from one layer to another when ready up to the data storage at the end. If performed very puristically, the data extractors wouldn't so much be extractors but APIs that are able to receive data from the appropriate sources. A list of advantages (green) and disadvantages (red) are listed in this section. req is related to  Extensibility of data gathering depends on  Data extraction technique - push or pull 	realized by Business rules engine representation API	depends on two separated servers Data extraction technique - push or pull

Figure 4.2: AK exploration and faceting in ArchiMind semantic wiki

Details and semantic interrelations of AK instances can be expanded in a tree-like fashion using '+' buttons (Label B). This shows how AK is interrelated to other AK. Requirement 'Compatibility' is expanded in the list (Label C) in Figure 4.2. Lists of AK instances can also be filtered based on keywords, as well as the classes and semantic relationships in Figure 4.1.

Label D shows how the list of requirements is faceted. Columns, each representing a facet, show the architecture elements and decisions that are related to the listed requirements via semantic relationships 'realized by' and 'depends on' in the ontology in Figure 4.1. Faceting allows users to view AK that has a certain relationship to the listed AK. Users can facet AK based on, e.g., related decisions, offered interface, and realized requirements.

File-based documentation content, e.g., from word processors and UML tools, and its layout is stored in wikipages (see Label E) using a WYSIWYG editor. 'Wikipage' is a class in the ontology in Figure 4.1 and instances of class Wikipage are used to store documentation content. ArchiMind allows for semantic annotation of phrases in documentation content that refer to AK instances, e.g., 'extractor' (an instance of AK type component) in the wikipage content in Figure 4.2 (see Label E). The annotated text on the wikipages is highlighted yellow and, when one clicks it, a pop-up menu shows the full description of the AK instance,

CHAPTER 4. ONTOLOGY-BASED ARCHITECTURE DOCUMENTATION APPROACH

its relationships to other AK instances, and to other wikipages that describe it. The annotation features are explained in more detail in Section 4.3.

The semantic annotations prevent issues with ambiguity, synonyms, homonyms, spelling errors, abbreviations, and context-dependent interpretation of AK in documentation content. This alleviates challenge (1) *Architecture documentation understanding*, described in Section 3.1.

When a fragment of text is annotated on a wikipage, a semantic relationship is created from the wikipages to the AK instance(s) that the annotated text fragment refers to, and vice versa. AK instances become traceable to the various fragments of documentation content (wikipages) that describe it and vice versa. For example, a users that clicks on requirement '*Compatibility*' shown in Figure 4.2 (Label C) will be able to see and navigate to wikipage '*24 Appendix - Push and Pull data*' (Label E) in which the requirement is annotated. The user can click on annotated text '*extractor*' on this wikipage to visit a description of this AK instance (component *extractor*). This helps users to locate (sources of) AK descriptions and thereby alleviates challenge (2) *Locating relevant architectural knowledge*, described in Section 3.1.

Semantic relationships in the ontology allows users to see what and how AK instances are interrelated, e.g., “a requirement is realized by components”, and thereby alleviates challenge (3) *Traceability*. If changes are made to an AK instance, e.g., a decisions is modified, a user can see what other AK might be affected, e.g., requirements depending on the decision. This alleviates challenge (4) *Change impact analysis*. Use of the ontology structure to check the existence of semantic relationships alleviates challenge (5) *Assessing design maturity*. For example, the correctness and completeness of an architecture can be assessed by checking if all requirements are *realized by* architecture elements and the buildability [9] of an architecture can be assessed by following the semantic relations that indicate dependencies between components.

Dublin Core [62] is used to store documentation meta-data, e.g., date, author, and version of documents. Next to the native version control of knowledge base instances in OntoWiki, also basic version control of wikipages was implemented in ArchiMind. This allows users to check whether documentation is up-to-date and can be trusted to reflect the AK in the running software project, thereby alleviating challenge (6) *Credibility of information*. The up-to-dateness of information is important for its credibility because software and architecture continuously evolve during a project and the documentation often lags behind.

The effort to maintain documentation, which is important for the adoption of a documentation approach, is also affected by the alleviation of aforementioned AK retrieval challenges. The presence of version and history information (to

alleviate challenge 6) also helps to see what documentation content is current during maintenance. Moreover, one can locate the documents in which AK has to be changed (challenge 2) and find related AK (challenge 3) that is affected by the changes made. This helps to prevent that a redundantly recorded AK description is only updated in one location during document maintenance. The semantic annotation of AK on wikipages introduces additional costs during maintenance, however, these can be minimized using an automatic annotation mechanism.

4.3 Annotating SA Documentation in ArchiMind

The retrieval of AK from file-based SA documentation suffers from issues with synonyms, homonyms, spelling errors, abbreviations, ambiguity, and context-dependent interpretation. Storing and annotating the content of software documentation in semantic wiki pages can alleviate these issues and supports AK retrieval. This section reports on the application of semantic annotation and knowledge retrieval using the ArchiMind semantic wiki system. Knowledge in documentation is annotated by indexing text with the lightweight software ontology in [104]. The process and the use-cases of this semantic annotation of software documents are described. The semantic annotation mechanism is illustrated by examples and use cases in the ArchiMind semantic wiki.

4.3.1 Semantic Annotation of Knowledge in Software Documentation

A WYSIWYG editor, with image upload functionality, was implemented to allow users to copy software documentation content in popular text editors and paste it in ArchiMind. Software specifications are stored as HTML in the content section of Wikipage instances of the ontology in Figure 4.1. The ontology contains Dublin Core [62] data properties to allow for specification of metadata (author, date, type, etcetera) for many possible sources of Software Engineering (SE) knowledge such as official documents, meeting notes, code snippets, interface specifications and e-mails.

Software documentation wiki pages are annotated in ArchiMind by indexing text to ontology instances with the following actions:

1. Select the text fragment that can be indexed to a SE knowledge instance of the ontology. Figure 4.3 depicts an example where the text "*order_config_mgr*" is selected.

CHAPTER 4. ONTOLOGY-BASED ARCHITECTURE DOCUMENTATION APPROACH

content **Config management in X2010**

The following requirements apply to `order_config_mgr`


Select the class instance that the text contains knowledge about below:

[Administration user interface](#)
[order configuration controller](#)

management should be part of the

`dc:publisher` `Klaas`
`rdf:type` `WikiPage`

`rdfs:label` `Config management in X2010 - Version 1`

Index selected text to knowledge of class/type: 

Architecture

NAVIGATE & CREATE	ANNOTATE
Behavior	Index to this class
Component	Index to this class
Architectural Structure	Index to this class
Setting	Index to this class

Figure 4.3: In-page annotation

2. Select the (sub)class in the ontology which contains the SE knowledge instance that the selected text should be indexed to. Ontology class selection is done in the in-page annotation menu. In Figure 4.3 this is class "*component*".
3. Select the SE knowledge instance, of the selected (sub)class, to which the selected text should be indexed. In the example in Figure 4.3 this is the instance with label "*order configuration manager*". It is assumed that an SE knowledge instance already exists. Otherwise it should be created before indexing.

Data from the indexing action above is stored in a separate annotation database table as (*URI: annotated Wikipage*), (*string: indexed text*), (*URI: SE knowledge instance*). When viewing a Wikipage, content of the Wikipage is checked against this database table and text that has been indexed is highlighted yellow. Clicking on the highlighted text will show the details of the SE knowledge instance(s) that the text is indexed to. This is further illustrated in the next section and depicted in Figure 4.4.

The indexing actions above are also used to annotate the Wikipage itself. A triple is stored in the knowledge base that captures the semantic relationship between Wikipage and the SE knowledge instance to which the text on the wiki page has been indexed to. The triple is stored as (*URI: annotated Wikipage*), (*URI: contains knowledge about <relation>*), (*URI: SE knowledge instance*). Another triple, creating a semantic relationship in the opposite direction, is stored as:

4.3. ANNOTATING SA DOCUMENTATION IN ARCHIMIND

The screenshot displays a web page titled "Config management in X2010" with a list of requirements. The requirement "REQ. 23 - X2010 should have a built-in order configuration management" is highlighted in yellow. A tooltip window titled "order_config_mgr" is overlaid on the page, showing details about the component. The tooltip includes a list of requirements it refers to (REQ. 23, REQ. 25, REQ. 24) and a table of knowledge instances.

Details on knowledge	
component	The order configuration controller (OCC) handles product-line specific order settings which enable or disable functions and behavior in X2010
description	
knowledge is	Component Specification X3 - Version 1
located in	Config management in X2010 - Version 1
rdf.type	Component
rdfs.label	order configuration controller

Figure 4.4: In-page knowledge retrieval

(*URI: SE knowledge instance*), (*URI: knowledge is located in <relation>*), (*URI: annotated Wikipedia*). The rationale for storing this triple is increased usability. Inverse relations can be shown in ArchiMind, however, this requires an extra action and knowledge of this feature.

4.3.2 Use of annotations in knowledge retrieval

Consider a software engineer who is interested in the requirements realized by component "order_config_mgr". A keyword search in ArchiMind on "order_config_mgr" returns the Wikipage, annotated in the previous section, with "order_config_mgr" highlighted yellow in the Wikipage content, as depicted in Figure 4.4. When clicking on the highlighted text, the indexed knowledge on a component with label "order configuration controller" and abbreviation "OCC", is shown. When properly defined, the SE knowledge instance contains semantic relationships to the requirements and behavior it realizes and the settings that impact it. The SE knowledge instance may also contain relations to other indexed wiki pages that have knowledge about it, but use an official, misspelled, abbreviated or synonymous name.

Also consider searching for "order" or "configuration" when these names are

CHAPTER 4. ONTOLOGY-BASED ARCHITECTURE DOCUMENTATION APPROACH

homonyms for classes, behavior, features, or functions. After semantic annotation the search results will include the correct SE knowledge instance and the Wikipage that have an index to the exact SE knowledge instance, e.g. a class. Wikipage instances listed in keyword search results contain expandable *<contains knowledge about>* relations to SE knowledge indexed in its content. These semantic relationships to and from the Wikipages, and the semantic relationships between SE knowledge instances aid users in knowledge retrieval.

4.4 Related Work

Several tools and approaches for managing AK exist such as ADDSS [18], Archium [52], AREL [102], PAKME [7], and SEURAT [15]. See [67] for an overview of AK management tools. These tools and approaches can be used to store, analyse, and retrieve formalized AK with semantics and they support many architecting activities. They differ from our approach in that they are not ontology-based (except for SEURAT [15]) and have less support for storing, managing, and retrieving knowledge contents stored in small and searchable chunks.

Happel and Seedorf [46] proposed documentation of Service Oriented Architectures (SOA) using Ontobrowse semantic wiki. A textual description is given of what typically should be included in an ontology for documenting SOAs, but no actual ontology is described. Their focus on SOA and lack of an ontology is a differentiation to our work.

Su *et al.* proposed KaitoroBase [95], a tool for exploring architecture documents, built on freebase semantic wiki. KaitoroBase allows for visualization and non-linear navigation of SADs stored in wikipages. A meta-model based on Architecture Driven Design is used, however, there are no details on whether other types of architecture documentation are supported. KaitoroBase provides exploration from a single node (say a single requirement), whereas our approach allows exploration from a set of related nodes (say all requirements realized by a component).

In Section 6.7 we discuss two other ontology-based SA documentation approaches together with their evaluation in industry.

4.5 Conclusion

In this chapter we described how an ontology can be used for organising and retrieving AK in SA documentation, and discussed similar usage of ontologies in related work. We introduced an ontology-based SA documentation approach which uses a lightweight software ontology and semantic wiki for AK retrieval. We described in detail how the ontology-based approach can address AK retrieval challenges identified in Chapter 3.

5

An Exploratory Study on Ontology Engineering for Architecture Documentation

This chapter illustrates how to build an ontology for SA documentation in a software project (RQ3). The usefulness of SA documentation depends on how well its AK can be retrieved by the stakeholders in a software project. Recent findings show that the use of ontology-based SA documentation is promising. However, different roles in software development have different needs for AK, and building an ontology to suit these needs is challenging. In this chapter we describe an approach to build an ontology for SA documentation. This approach involves the use of typical questions for eliciting and constructing an ontology. We outline eight contextual factors, which influence the successful construction of an ontology, especially in complex software projects with diverse AK users. We tested our 'typical question' approach in a case study and report how it can be used for acquiring and modeling AK needs.

5.1 Introduction

We built an ontology for the SA documentation in a software industry project, in order to implement ontology-based SA documentation. To do so, we had to consider what ontology engineering approach would be suitable. This chapter is about building the ontology structure for ontology-based SA documentation. It is not about instantiating (or 'populating') this ontology or about the experiment reported in Chapter 6, in which the ontology-based approach is evaluated.

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

Developing an ontology for a software project in industry should be economically feasible, i.e. it should be efficient and accurate, for organization and individual users [47]. If the ontology is inaccurate, documentation users will not retrieve or understand the AK that they need. Users would lose interest and confidence in ontology-based SA documentation, and revert to other means to get the required AK.

In the context of large software projects it can take much time and effort to develop an accurate ontology. Knowledge acquisition from many diverse stakeholders, each having their own needs and views [9] of AK, is required to build an accurate ontology. These users of AK are generally pressed for time and their primary interest is seldom about making documentation. Moreover, the AK needed by users in large software projects is often domain specific and complex.

Developing the ontology is not a one-time effort because the AK needs of SA documentation users shift over time. For example, during development users will be interested in AK that relates requirements to the components implementing those requirements, while during integration testing (other) users might be interested in relations between software releases and requirements coverage. Shifting AK needs necessitates a regular evaluation and revision of the constructed ontology.

In this chapter we describe eight contextual factors in software projects. These contextual factors influence ontology engineering for SA documentation, especially in large and complex projects. We devised a 'typical question' approach for ontology construction that takes the contextual factors in large and complex software projects into account. In this approach typical questions about AK are acquired from SA documentation users and used to build an ontology. These typical questions are frequently asked by AK users¹ during their everyday tasks, i.e. questions that represent their everyday AK needs.

We applied the 'typical question' approach in an exploratory industrial case study which provided several insights. In the case study we explored how well our 'typical question' approach was applied to construct a useful ontology by acquiring and modeling AK needs of many diverse users that use SA documentation in different projects and product lines. The 'typical question' approach can continuously refine the AK ontology when AK needs evolve.

This chapter is motivated by the lack of applied ontology engineering approaches for constructing an ontology for SA documentation. We make the following contributions:

- Illustrate a 'typical question' approach for constructing the ontology used in ontology-based SA documentation.

¹We consider AK users the same as SA documentation users in this work.

- Outline important contextual factors that influence ontology engineering for SA documentation in software projects.
- Demonstrate how the 'typical question' approach can be applied through an exploratory case study in a software industry project.

This chapter is organized as follows. Background of ontology-based SA documentation, ontology engineering, knowledge acquisition, and Grounded Theory is given in Section 5.2. Section 5.3 describes our 'typical question' approach for ontology construction. Section 5.4 describes the contextual factors that influence ontology engineering for SA documentation in software projects. Section 5.5 reports the exploratory case study and the lessons learned from it. Section 5.6 presents related work and Section 5.7 concludes this chapter.

5.2 Background

5.2.1 Ontology Engineering for SA Documentation

The ontology used in ontology-based SA documentation determines what AK one can retrieve using its structure and semantics. One can use a predefined ontology (as in [45]) or build an ontology specifically for a software project domain. A domain specific ontology can be built, e.g., by letting document authors and architects identify AK concepts in existing SA documentation [51].

The approach proposed by Jansen *et al.* in [51] is, to our current knowledge, the only approach aimed at deriving a domain specific ontology for ontology-based SA documentation. Their use of existing documentation however has several limitations:

- Existing SA documentation might not be available.
- Existing SA documentation might not convey the AK needed by all SA documentation users.
- Experts that identify AK concepts from SA documentation might not know the AK needs of all users.

Building and maintaining an ontology for a certain domain (such as in this case the domain of AK) is called ontology engineering. An overview of the ontology engineering phases, adapted and simplified from [87], is depicted in Figure 5.1. Arrows in this figure represent a transition to another phase.

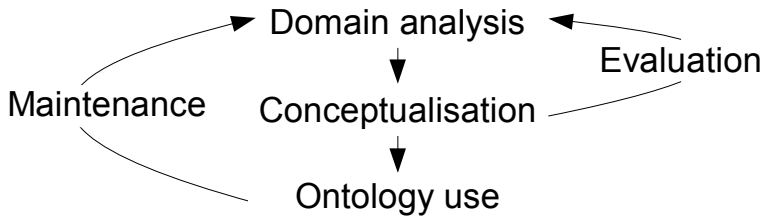


Figure 5.1: Ontology engineering phases

The process starts with the domain analysis phase which includes identification of the domain scope and the AK needs of users of SA documentation.

AK concepts are derived from AK needs and used to build a domain ontology in the conceptualization phase. The quality of the constructed ontology is then evaluated in the evaluation phase. Domain analysis, conceptualization and evaluation is iterated until a satisfactory ontology is obtained.

After construction the ontology is used to annotate and retrieve AK from SA documentation. The ontology is maintained to cope with evolving AK needs and concepts.

5.2.2 Knowledge Acquisition

Building an AK ontology requires knowledge acquisition during the ontology engineering phases described in the previous section. Knowledge acquisition is part of building knowledge-based systems in general. The suitability of knowledge acquisition approaches differs per domain. A separation can be made between top-down, middle-out and bottom-up approaches.

Top-down approaches start modelling based on a general model, with extension and refinement to suit specific needs. These approaches can be efficient and accurate for well-defined domains in which an ontology engineer knows quite well about the general concepts, questions, tasks, and use-cases. Bottom-up approaches are used to build a model based on specific domain knowledge. These are suitable for domains in which an ontology engineer cannot predetermine all the concepts, questions, tasks, and use-cases. Such domains can be broad, multi-disciplinary, or novel. Middle-out approaches combine a top-down and bottom-up approach. These work well for domains that are partially well-defined and partially specific.

5.3. ONTOLOGY ENGINEERING USING THE 'TYPICAL QUESTION' APPROACH

5.2.3 Grounded Theory

Our 'typical question' approach makes use of coding techniques from Grounded Theory in which a domain theory is generated by empirical generalization [40]. This is a bottom-up approach to knowledge acquisition (see previous section 5.2.2).

First, data is collected in the domain under investigation, and patterns that indicate concepts are identified from this data. The identified concepts are aggregated into categories. Second, properties of the identified categories are developed by constantly comparing the categories with collected domain data. These properties are developed with respect to a 'core' or 'central' category [93] that is the subject of investigation in the domain. Memos capture thoughts about the possible concepts, categories, and relationships in the domain data. Finally, if no new properties can be identified from domain data (categories are 'saturated'), the domain model is compared to literature.

Urquhart *et al.* discuss several myths about Grounded Theory in [109] and conclude that Grounded Theory is a rigid and flexible method that is suitable for use in information systems research. Grounded Theory is able to generate theories that are relevant to practitioners [3] and that are 'grounded in data' [40].

Grounded Theory can be used during ontology engineering when analysing a domain and constructing an ontology. An ontology engineer starts with the collection and analysis of domain data in the domain analysis phase (See Figure 5.1 and Section 5.2.1). Similarly, data is collected and analyzed in a domain of interest when one uses Grounded Theory in the domain analysis phase. In the ontology conceptualization phase an ontology engineer can use Grounded Theory to create classes (or 'categories') and relationships between classes.

5.3 Ontology Engineering using the 'Typical Question' Approach

We devised an ontology engineering approach that uses typical questions to create an ontology in the context of a large and complex project at Océ Technologies.

The 'typical question' approach focuses on the elicitation of questions that AK users normally try to answer during their use of SA documentation. We want to understand what AK users typically ask of the SA documentation. Knowing the detailed knowledge needs of AK users allows one to create SA documentation that is optimal for its users and therefore cost-effective [31].

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

The typical questions convey what concepts and relationships AK users want to retrieve from SA documentation. An ontology engineer identifies these concepts and relationships on basis of 'typical questions' to construct an ontology. This ontology is subsequently used to support retrieval of the concepts and relationships from SA documentation.

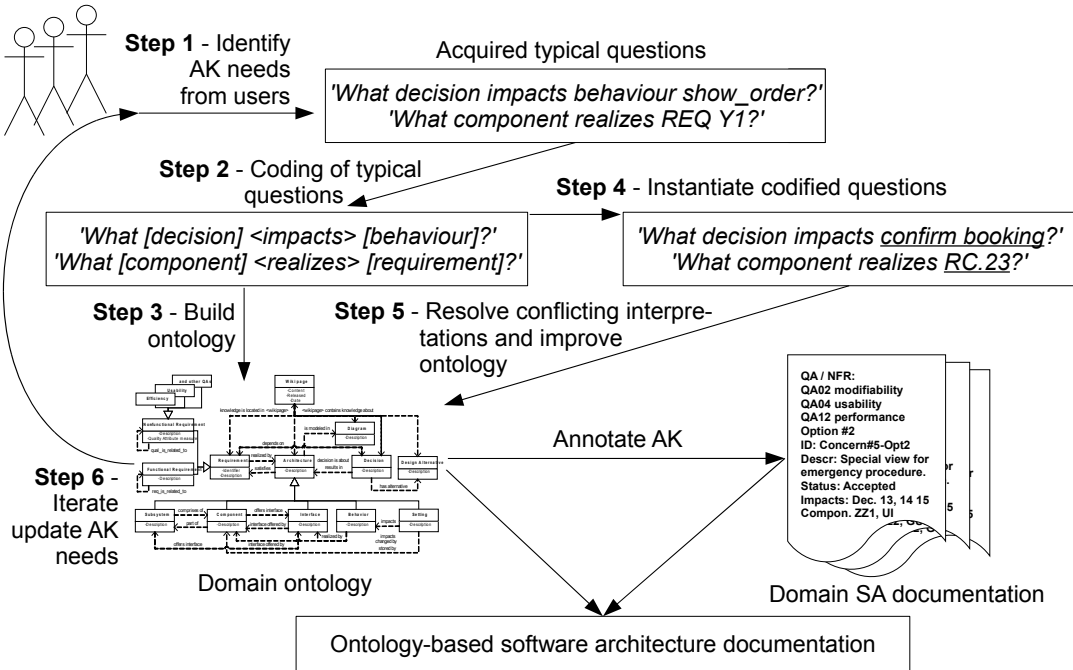


Figure 5.2: Overview of the proposed 'typical question' approach for ontology engineering

Figure 5.2 depicts an overview of our approach. In this approach the process of creating an ontology requires one or more steps in each ontology engineering phase.

In the domain analysis phase of ontology engineering, an ontology engineer acquires typical questions from documentation users to capture their AK needs (step 1). In the conceptualization phase these typical questions are codified (step 2) and modeled in an ontology (step 3) using coding mechanisms from Grounded Theory. In the evaluation phase typical questions are instantiated (step 4) to resolve conflicting interpretations of AK concepts and to evaluate and improve the constructed ontology (step 5).

5.3. ONTOLOGY ENGINEERING USING THE 'TYPICAL QUESTION' APPROACH

In the ontology use phase, existing file-based documentation is annotated and the semantic wiki is used for AK retrieval. During the maintenance phase the techniques to acquire the typical questions (in step 1) are iterated to detect changes of AK needs and if the ontology needs to be updated (step 6).

Interviews, daily logging of typical questions, and mailing lists are techniques that can be used to acquire typical questions. These are further detailed in Section 5.4.1.

5.3.1 Domain Analysis Phase (Step 1)

In the first step of our approach AK needs are identified from all users, or a representative subset of users, by acquiring their typical questions about AK. These are questions that are representative of what AK users ask themselves during their everyday activities, e.g., during design, development, quality assurance, testing etc. For example, a developer might ask: “*what components and behavior are impacted by my change in this subsystem?*”. Snippet 1 gives an example of two typical questions.

Snippet 1 - Example of typical questions

- “Which module assures adherence to requirement ‘24 - admin login?’”
- “What quality attributes are realized by subsystem ‘transaction handler?’”

Normally such typical questions are answered by reading architecture and design documents, inspecting source code, or consulting colleagues. Instead we use these typical questions in our approach to build an ontology which in turn provides users with structure and semantics to answer these typical questions from ontology-based SA documentation.

A core idea behind the use of typical questions is the assumption that the AK needed in any use-case, scenario, or task can be accurately represented as a set of questions that should be answered. As such, acquiring sets of typical questions allows for a fine-grained and detailed specification of the AK needed for tasks, use-cases, and scenarios. Recording or recalling these typical questions can reduce difficulties for users in articulating domain knowledge [107], i.e. increase efficiency by reducing effort. This addresses difficulties in acquiring AK needs of diverse users in a complex and multi-disciplinary domain.

Benefits can be gained by the use of typical questions that are 1) efficiently acquired from many users 2) tangible for users, 3) accurately represent AK needs and daily practice of users, 4) do not require extensive abstract thinking from

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

users when acquired, yet 5) convey much conceptual information, and 6) can be used throughout the ontology engineering phases. These benefits help to minimize the effort required from individual users of SA documentation, as well as the total effort required in complex domains with many diverse users.

5.3.2 Conceptualization and Evaluation Phase (Steps 2-5)

In the ontology conceptualization phase an ontology engineer identifies AK concepts from typical questions in order to build an ontology. AK concepts are identified from phrases, i.e. one or more words, in the acquired typical questions. Phrases in typical questions are classified as representing a class, relationship, or attribute in the ontology. This is done by applying coding techniques from Grounded Theory, discussed in Section 5.2.3.

An ontology engineer starts with open coding in which textual data, a typical question in this case, is broken down into discrete parts, e.g., words and phrases. These discrete parts are examined in detail and classified (or 'categorized'²) by comparing data for conceptual similarities and differences [93]. Snippet 2 gives examples of typical questions in which open coding, or 'labeling', i.e. assigning conceptual names (in *italic* between square-brackets), is applied to phrases that refer to AK concepts. Various labels are applied to illustrate possible conceptualizations in open coding. This process corresponds to the first phase of the overall process of Grounded Theory described in Section 5.2.3.

Snippet 2 - Example of initial open coding of typical questions

- “Which module [*module, component, hardware, subsystem*] assures adherence to [*constraints, satisfies, realizes*] requirement ‘24 - admin login’ [*functional or non-functional requirement, Quality Attribute*]?”
- “What quality attributes [*quality attribute, non-functional requirement*] are realized by [*constraints, satisfies, realizes*] subsystem ‘transaction handler’ [*subsystem, component, module*]?”

Axial (or 'theoretical') coding is then used to (re)assemble and interrelate the classes, identified during open coding, to their sub or superclasses. Selective coding is then used to integrate, interrelate and refine the classified concepts in an ontology based on the central 'theme', 'idea', or 'category' [93] of the

²The term 'category' is used in Grounded Theory literature, however we adopt the term 'class' instead of 'category' for consistency with the rest of the chapter.

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

is used for a concept in many typical questions, we cannot assume based on this frequency that all users have the same interpretation of this concept. Users might mean different things with the same phrase.

An ontology engineer evaluates the interpretation of AK concepts by presenting users with different instantiations of typical questions that have been coded and classified. Phrases in these typical questions that have been coded to an ontology class or relationship are replaced with different instances of the same class or relationship, e.g., instances '*requirement X1*', '*R2 - admin login*', and '*Req. 3*' for class requirement. This is akin to validation in Grounded Theory where respondents in a study comment on how well the theory, e.g., represented as a 'story', fits their case [93].

An ontology engineer subsequently asks users feedback about the representativeness, correctness, and accuracy of these typical questions with different instances of AK concepts. This supports identification of conflicting interpretations of AK concepts (step 5) between users working in diverse roles and disciplines. Snippet 4 gives the typical questions from Snippet 3 with different instances of AK concepts.

Presenting typical questions with different instances of AK concepts allows users to evaluate the concepts in different contexts. The interpretation of an AK concept may differ between AK users when they are presented with different instances of the AK concept. Part of the evaluating users may remark that "*allow backup scheduling*" (in Snippet 4) is a non-functional requirement or feature instead of a functional requirement. This identifies incorrect or inaccurate classification of AK concepts in typical questions.

Snippet 4 - Example typical questions with instantiated AK concepts for evaluation

- "Which component satisfies functional requirement '*18 - allow backup scheduling*'?"
- "What non-functional requirements are realized by subsystem '*Order configuration*'?"

Grounded Theory coding allows an ontology engineer to construct an ontology "grounded in data" with relative high efficiency and accuracy and without requiring extensive effort from documentation users. This requires minimal guidance by domain experts which might not be readily available in many software projects.

After the ontology is conceptualized an ontology engineer may decide to compare the ontology to other ontologies, e.g., in literature. This allows identification of

5.4. CONTEXTUAL FACTORS IN ONTOLOGY ENGINEERING

useful domain-independent AK concepts that may be missing in the constructed domain specific ontology.

5.3.3 Ontology Maintenance Phase (Step 6)

When AK needs shift and concepts evolve the ontology has to be maintained to remain accurate. The ontology engineer can identify a shift in AK needs by re-acquiring typical questions (step 1). An ontology engineer then compares the newly acquired typical questions and those previously acquired. If AK needs have significantly shifted, steps 2 through 5 of the approach can be repeated to update the ontology. Changes in AK concepts are detected in step 4 of the approach when newly acquired typical questions are instantiated and evaluated by AK users.

The necessity of reacquiring typical questions should be estimated or planned at the start of each software project phase. Estimating the necessity of reacquiring typical questions or the required accuracy of this reacquisition is outside the scope of this chapter.

5.4 Contextual Factors in Ontology Engineering

In the introduction section of this chapter we briefly described how the development of an SA documentation ontology is affected by its users and the software project context. These Contextual Factors (CF) influence the use of the 'typical question' approach. In agile software development, we learn that contextual factors influence the successful adoption of agile practices [61]. We recognize that these factors are also applicable in terms of producing SA documentation.

Production of SA documentation takes place in the context of software development projects and the context of a project and its products is captured in SA documentation. Moreover, users of SA documentation in a project determine what AK is relevant to capture in SA documentation. Several characteristics of the SA documentation users influence how one can most effectively find out what AK is relevant to these users.

An ontology engineer would need to consider these factors whilst building a suitable ontology for SA documentation:

CF1 Number of AK users - impacts on the extent of AK needs acquisition. For example, interviewing a large number of AK users might be infeasible in a project.

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

- CF2 Accessibility of AK users - communication with AK users can be constrained by their accessibility, e.g., in terms of location and schedule.
- CF3 Commitment of AK users - influences how much time and effort users are willing to spend on providing their AK needs and ontology evaluation.
- CF4 Diversity of AK users - the role, experience, and education background can impact on the interpretation of AK concepts between users.
- CF5 Product domain complexity and specificness - influences the effort required of an ontology engineer to understand and model AK concepts in a software product domain.
- CF6 Product domain multidisciplinary - impacts on how many different AK concepts, e.g., from the healthcare, embedded systems, and chemistry discipline, are needed by AK users.
- CF7 Shifting AK needs - Users provide their AK needs based on their roles and the current tasks. When a project progresses and their tasks change, these AK needs may shift. Additional AK needs may be required to enhance an ontology.
- CF8 Volatility of AK concepts - AK concepts can change over time. This affects the accuracy of the AK concepts initially captured in an ontology.

5.4.1 Contextual Factors Influencing the Acquisition of Typical Questions

During the acquisition of typical questions, we noticed that contextual factors influenced what acquisition techniques we used. We have used three knowledge acquisition techniques (see Table 5.1) in our case study. We summarize, based on our experience, how contextual factors 'user accessibility' and 'user commitment' influence the performance of each acquisition technique.

Table 5.1: Evaluation of techniques for acquiring typical questions

Technique for acquiring typical questions	Suitable for Contextual factors	Results in
Interviews	high user accessibility and commitment	High accuracy and low efficiency
Daily log	medium user accessibility and commitment	Medium accuracy and efficiency
Mailing list	low user accessibility and commitment	Low accuracy and high efficiency

5.4. CONTEXTUAL FACTORS IN ONTOLOGY ENGINEERING

Interviews with AK users allow an ontology engineer to clarify their responses and thoughts, e.g., using examples, and this allows an ontology engineer to acquire AK needs. Interviews however require high commitment, accessibility, and much time and effort from users and ontology engineer. Relatively high accuracy is traded off against lower efficiency.

A daily log, in which users consistently record their typical questions each day, requires a fair amount of accessibility to and commitment of users. Even though the time-efficiency and required effort is better than that of interviews, it is less accurate due to the lack of direct interaction with an ontology engineer. Accuracy becomes even lower when user commitment is low and AK needs are not consistently recorded every day.

The use of a mailing list provides time-efficient and effortless acquisition of typical questions, even with low user commitment and accessibility. The accuracy of AK needs acquisition is low as there is no direct interaction between documentation users and ontology engineer, however, more AK needs can be acquired. The use of a mailing list can prove to be very suitable for projects with many users and distributed development. Bürger *et al.* provide evidence in [16] that suggests that the use of e-mail is more efficient than face-to-face meetings.

The use of multiple acquisition techniques at the same time may be suitable in some situations. For example, interviews and a mailing list may be used at the same time for acquiring typical questions from a group of practitioners residing in the same location as the ontology engineer and another group of practitioners working elsewhere.

5.4.2 Contextual Factors in the Domain Analysis Phase

During domain analysis an ontology engineer identifies what AK is needed by SA documentation users. If AK needs are overlooked documentation users will have less support to retrieve this AK. The possibilities for acquiring AK needs from users and the amount of AK needs in a software project influence how much time and effort is required from the ontology engineer and AK users for the identification of AK needs.

Large software projects typically have many (CF1) different (CF4) stakeholders that are users of AK, such as software architects, engineers, testers, and product managers. AK in large projects is often multidisciplinary in nature (CF6), conveying views from many diverse stakeholders who are both users of AK and experts in their respective domains (CF4).

We need to talk to a lot of AK users before all the AK needs for their roles (CF4)

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

are clear, especially in complex software projects (CF5). This becomes difficult when AK users are not accessible (CF2) or committed (CF3) to cooperate [107] [47].

In our approach typical questions are used to capture domain-specific AK needs from many diverse users. Formulating questions requires relatively little effort from users and acquiring questions can be scaled up in large projects. The accessibility and commitment of AK users is addressed by selecting one of several techniques for acquiring typical questions listed in Table 5.1.

5.4.3 Contextual Factors in the Conceptualization and Evaluation Phase

After AK needs are identified, AK concepts can be derived from the AK needs and modeled in an ontology. If the ontology is inaccurate or ambiguous it will not support efficient and effective AK retrieval.

Conflicting interpretations of AK concepts are likely to occur with many users [107] (CF1) with diverse roles (CF4) that work from different disciplines (CF6). An ontology engineer might not have complete and thorough knowledge of all AK concepts and their interpretation in a software product domain that is very specific and complex (CF5). Reuse of a generic ontology is limited in such domains. Domain experts can help with ontology modeling and evaluation, yet these experts may be inaccessible (CF2) or uncommitted (CF3).

In our approach an ontology engineer uses coding techniques from Grounded Theory to model AK concepts from typical questions. This coding process is refined and iterated to improve the accuracy of ontology modeling without heavily relying on domain experts. Typical questions are then evaluated by AK users to verify that AK concepts in the ontology are accurate and interpreted consistently between AK users. The accessibility and commitment of AK users is addressed by selecting one of several techniques listed in Table 5.1 for acquiring evaluations.

5.4.4 Contextual Factors in the Maintenance Phase

The AK retrieved by users should remain accurate even when their AK needs evolve (CF7) and AK concepts evolve (CF8). Therefore the ontology should be updated accordingly [98]. Updating the ontology should be efficient 1) for economic feasibility and 2) to prevent of any lag between the moment AK needs and concepts change and this new AK can be retrieved by users [47].

5.4. CONTEXTUAL FACTORS IN ONTOLOGY ENGINEERING

Table 5.2: Influence of Contextual Factors in Ontology Engineering Phases

All Ontology Engineering Phases	
Contextual Factors	Influence
CF2: Accessibility of AK users CF3: Commitment of AK users	Accurate acquisition of AK needs becomes difficult when AK users are inaccessible or uncommitted. Ontology construction and evaluation needs involvement of committed AK users.
Domain Analysis Phase	
CF1: Number of AK users CF4: Diversity of AK users CF5: Product domain complexity and specificity CF6: domain multidisciplinary	These CFs influence how many different AK concepts have to be identified, supporting the AK needs of users in various roles, disciplines, and product domains.
Ontology Conceptualization and Evaluation Phase	
CF1: Number of AK users CF4: Diversity of AK users CF5: Product domain complexity and specificity CF6: domain multidisciplinary	Conflicting interpretations of AK concepts are likely to occur when there are many diverse AK users, each having their own jargon in their domain and role. An ontology engineer may not have thorough understanding of all disciplines and product domain specific concepts, and require assistance from domain experts.
Ontology Maintenance Phase	
CF7: Shifting AK needs CF8: Volatility of domain concepts	AK needs of users shift, e.g., between software project phases and when domain concepts change. Maintenance may be needed to keep the ontology up to date with the AK needs of its users.

AK needs shift and AK concepts can become deprecated in a software project when architecture, design and development methods change, societal and organizational changes impact the SA [9], new insights and solutions are found, project phases progress, or when concepts from the product domain(s) evolve. This is likely to happen in complex project domains (CF5) involving many (CF1) diverse (CF4) users working in multiple disciplines (CF6) that evolve with time. In our approach typical questions are re-acquired and compared to previously acquired questions to detect shifting AK needs and AK concepts.

5.5 Case Study

In this exploratory case study [82] on the use of the 'typical question' approach we explore two questions:

- *How do the contextual factors in this case study influence the application of the 'typical question' approach?*
- *How well does the 'typical question' approach work in this case study to construct a useful ontology for SA documentation?*

We developed an ontology that was applied to SA documentation at Océ Technologies, an international leader in digital document management and a Canon Group company. This SA documentation specifies the software for document printing machines developed at Océ Technologies and is used in multiple projects and product lines.

The documentation for which the ontology was built consists of 7 SA documents with 79 pages in total and is a small yet representative subset of the available types of documents in a project at Océ Technologies. This subset of documents was selected because of timing constraints for the experiment reported in [30].

The products built at Océ Technologies evolve with market needs, which introduces a high rate of change (CF8) in their domain. Their product teams deal with software, firmware and specific hardware (CF5, CF6) and consist of diverse documentation users (CF4) such as domain architects, product testers, workflow architects, etc. Océ applies agile development in a product line environment. This means that the architectural design and software project phases iterate rapidly (CF7).

A software professional at Océ estimated that there are well over 50 users of the SA documentation (CF1). Software projects typically take place in three locations in Europe. However, our study was limited to documentation users in one site (CF2). Even though documentation users were pressed for time, they were committed to this research project (CF3).

5.5.1 Domain Analysis Phase (Step 1)

In order to work with the schedule and accessibility of the AK users at Océ Technologies, we used a mailing list to acquire their AK needs. We asked SA documentation users to send their typical questions about AK that they had during their work activities. 7 documentation users, among which software engineers, a software project manager and a software architect, provided 17 questions.

Snippet 5 below gives a subset of the acquired typical questions at Océ. Parts of these questions are obfuscated for confidentiality reasons.

Snippet 5 - Subset of typical questions acquired at Océ

- “What is the rationale behind this requirement? (And whom can we ask?)”
- “Which subsystem is responsible for fixing the XX defaults based on the device configuration?”

AK needs acquisition during domain analysis using a mailing list was time- and cost-efficient. Because many users involved in this case study proved to be accessible and committed the acquisition was, in retrospect, also relatively accurate (see Table 5.1 for comparison).

5.5.2 Conceptualization and Evaluation Phase (Steps 2-5)

After acquisition of typical questions in the Océ case study we labeled and classified the AK concepts in these typical questions. Snippet 6 lists three typical questions, two from Snippet 5, that we acquired. Labels (between square brackets) show the classification of phrases and words after we applied the coding mechanisms from Grounded Theory [93].

Snippet 6 - Coding of typical questions at Océ

- “What is the rationale [*Decision (class)*] behind [*depends on (relationship)*] this requirement [*requirement (class)*]”? (And who [*stakeholder (class)*] can we ask?)”
- “Which subsystem [*subsystem (class)*] is responsible for fixing [*changed by (relationship)*] the defaults based on the device [*Device (class)*] configuration [*setting (class)*]?”
- “I changed the behavior [*behavior(class)*] of some interface method [*method(class)*] after I had convinced myself that the method [*method (class)*] was not used at all in other parts of the system and so the change would have no [*change task (class)*] impact [*impacts (relationship)*]. However, at the same time another team made functional enhancements [*change task (class)*] that relied [*depends on (relationship)*] exactly on the *old* behavior [*behavior(class)*] of that method [*method(class)*]. Could I have known that a new subsystem [*subsystem(class)*] dependency [*depends on (relationship)*] on this

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

method [“*method (class)*”] was upcoming [“*change task (class)*”] or was it just bad luck that these actions [“*change task (class)*”] had crossed each other?”

During coding we found that many words and phrases in the typical questions could almost directly be translated into relationships and classes, e.g., similar to Figure 5.3. We recorded the rationale for our actions during the coding steps in a document (named ‘field notes’ or ‘memos’ in Grounded Theory). We instantiated the classified AK concepts in typical questions, of which two are shown in Snippet 7.

Snippet 7 - Typical questions with instantiated AK concepts at Océ

- “What decision depends on requirement ‘*REQ. 23*’?”
- “Which subsystem changes setting ‘*external indication light color*’?”

Feedback interview sessions were held with a small group of committed and accessible documentation users in diverse roles. These documentation users evaluated the classified and interrelated AK concepts and resolved conflicting interpretations. A software designer and software engineer each evaluated 7 questions with instantiated AK concepts on their accuracy as well as their relevancy and representativeness for the roles in the project. A software architect provided feedback on the interpretations of several AK concepts. These interviewees did not partake in providing the typical questions. Below is a summary of how a conflict between the interpretations of ‘behavior’ and ‘feature’ was resolved:

The AK concepts ‘behavior’ and ‘feature’ are on different abstraction levels and used by users in different roles: ‘Feature’ is similar in meaning to ‘behavior’, but is a term adopted by users that work from a business perspective.

‘behavior’ is adopted as the primary representation of the AK concepts in the ontology.

Figure 4.1 depicts an ontology partially built using the coded and evaluated AK concepts from Snippets 6 and 7. This ontology was used for the industrial experiment reported in Chapter 6. Note that we only included AK concepts in this ontology that could later be annotated in the subset of documents used in the experiment. We did not include the other identified concepts such as ‘Change task’, ‘Method’, ‘Device’, and ‘Stakeholder’.

We compared the identified AK concepts and relationships to those in the Light-

weight Software Ontology proposed by Tang *et al.* in [104]. The Lightweight Software Ontology offers a starting point or a template to help ontology engineers capture AK that is commonly used. The AK concepts and relationships we identified largely coincided with those in the Lightweight Software Ontology, i.e., the 'typical question' approach was also able to identify commonly used domain-independent AK.

We adopted concepts 'Wikipage' and 'Diagram', derived from concept 'DC' (Dublin Core) in the Lightweight Software Ontology, to enable ontology-based documentation in a semantic wiki tool. The steps taken in the construction of the ontology in Figure 4.1 are part of a middle-out approach (see Section 5.2.2), combining a top-down predefined ontology (i.e. the Lightweight Software Ontology) and bottom-up acquisition of domain specific AK needs using the 'typical question' approach. 11 semantic relationships and 4 classes, appended with "(Océ)", in the ontology are Océ domain specific. Concepts 'Behavior' and 'Setting' in this ontology have specific semantics in the Océ product-domain.

The ontology was used to store the content of the file-based documentation subset in the semantic wiki as wikipages. The ontology classes and relationships were instantiated by semantic annotation of AK in the SA documentation content in wikipages.

The semantic annotation of AK concepts, i.e. the documentation content itself, described above is not part of our 'typical question' approach. We describe this to illustrate how the constructed ontology was used as part of an ontology-based documentation approach.

5.5.3 Ontology Maintenance Phase (Step 6)

In the Océ case study we observed that the AK needs of users shifted between software project phases. High-level AK about product lines, changing independent of individual product projects, is present in the documentation and can introduce shifts in AK needs. Moreover, much product research and innovation takes place in the R&D department of Océ, e.g., on hardware and mechanical components. This introduces volatility of AK concepts.

Several months after previous steps were executed and ontology-based SA documentation was implemented, we asked a diverse group of documentation users to record their typical questions about AK in a daily log (technique from Table 5.1). We decided to use daily logs to acquire typical questions because users were still quite accessible and committed and because it gave a balanced trade-off between accuracy and efficiency of AK needs acquisition. We acquired 39 typical

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

questions from 9 documentation users, including 5 software engineers, 2 product testers, 1 software architect and 1 project manager.

We collected daily logs during the build and integration phases of the project in which ontology-based SA documentation was implemented. New AK needs were acquired compared to the known AK needs acquired at an earlier project phase. The daily logs contained typical questions about software builds, releases, planning, product-lines, control flow, physical products, and automated tests. Considerably fewer typical questions about change impact were acquired as compared to the typical questions initially collected. Four examples of typical questions containing new AK needs are given below in Snippet 8.

Snippet 8 - Newly acquired typical questions at Océ

- “I need to find AK on build XX from a different domain team (but for the same product).”
- “Is setting XX for device YY useful for product ZZ in product line?”
- “How is development tool XX used in offshore location YY?”
- “Is state XX persistent? A test case fails on behavior YY.”

5.5.4 Lessons Learned

The case study gave us insight in how contextual factors influenced the application of the ‘typical question’ approach in the Océ project and whether the approach could be used to construct a useful ontology.

- *How do the contextual factors in this case study influence the application of the ‘typical question’ approach?*

The ‘typical question’ approach was used to acquire AK needs for many diverse AK users (CF1 and CF4) in steps 1 and 6 of the approach. The roles of these AK users include software engineers, architects, project managers, and product testers. The time and effort spent by the AK users was acceptable for them as they had spent around 5-10 minutes to type an email with typical questions or a few minutes each day to record a typical question in their daily log.

AK users were not only able to phrase their own typical questions in steps 1 and 6, but also evaluate the questions of other AK users during steps 4 and 5 of the approach. Part of the typical questions acquired in the case study were not only about AK but also about detailed design and implementation details. We

observed that the amount of AK needed by AK users is influenced by their role (CF4) and the software project phase in which typical questions are acquired.

In the case study the researchers had good access to AK users who were quick to help. AK users provided typical questions, clarified AK needs, and evaluated the ontology. This commitment also impacted the time and effort spent by AK users to phrase and rephrase their typical questions, making sure they were relevant, and consistently record them in a daily log. Accessibility (CF2) and commitment (CF3) of AK users are regarded as preconditions for the 'typical question' approach to work. However, our evaluation (see Table 5.1) suggests that our approach can address low accessibility by using an acquisition technique involving email or mailing lists. A case study in a project where SA documentation users have low accessibility would give insight in this, e.g., in distributed development.

The diversity of AK users in terms of their roles (CF4) and disciplines (CF6) was relevant and necessary for understanding and aligning diverse AK concepts. Examples of this was the interpretation of 'setting' between the tester and developer role and the interpretation of 'feature' between users from engineering and business disciplines. These conflicting interpretations were detected and resolved using instantiated typical questions in steps 4 and 5 of the approach. As such, it appears advantageous to have stakeholders in different roles to participate in evaluation.

We found that AK concepts that are specific to the product domain (CF5) do not always show up in acquired typical questions. The reason is that we asked the AK users to provide their typical questions about architecture, and consequently the users did not actively ask questions about background knowledge on the product domain. Part of the product domain specific AK was implicit or omitted in the typical questions. Therefore this knowledge was not explicit in the constructed AK ontology.

Several months after the initial ontology was constructed, the approach was used again to detect shifting AK needs in the same project (CF7). We did observe shifting AK needs, but the meaning of the classes and relationships remained the same (CF8). The typical questions that we acquired from users in the initial execution of step 1 in the approach did not cover the AK needed by users for their later tasks. This means that if user tasks change frequently, more ontology maintenance is needed.

- *How well does the 'typical question' approach work in this case study to construct a useful ontology for SA documentation?*

In a questionnaire we asked five Océ professionals that are users of AK to evaluate the ontology. This evaluation took place after they used the ontology to retrieve

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

AK from SA documentation in the experiment reported in Chapter 6. Table 6.5 reports questionnaire results. We asked the five AK users whether *the ontology is a correct representation of reality*. Three AK users answered “yes” and two AK users answered “to a certain extent”. These two AK users remarked that the ontology should include more printing machine domain knowledge.

Not all of the printing machine domain knowledge that was identified was included in the evaluated ontology depicted in Figure 4.1. This is because the evaluated ontology was built for annotating a subset of SA documentation. AK concepts that were not described in this subset of SA documentation were not included in the evaluated ontology. Moreover, we asked AK users to provide typical questions about the architecture, not about the product domain. As a result we cannot claim that our approach is comprehensive for building a full ontology for all possible AK needs.

All five Océ professionals confirmed in the questionnaire that they found it practical to work with the ontology. Moreover, all five Océ professionals found the ontology helpful to reason about what AK is in SA documentation, and what AK should be in the SA documentation. This gives us confidence that the ‘typical question’ approach worked well to construct a useful ontology in this case study.

The ontology constructed in the case study was used together with the ArchiMind semantic wiki to construct ontology-based SA documentation for the experiment reported in Chapter 6. 26 Océ professionals used this ontology-based SA documentation to retrieve AK and their efficiency and effectiveness was significantly higher than when using file-based SA documentation to retrieve the same AK.

5.6 Related Work

In this section we discuss how aspects of the ‘typical question’ approach for ontology engineering relate to aspects of other ontology engineering approaches and how they differ.

Jansen *et al.* describe how they construct an ontology from existing SA documentation in [51]. Researchers and a software architect identified and annotated classes of AK in documentation text, to elicit the AK needs in a documentation use case about architectural reviews. Existing SA documentation and the use case(s) may however not convey AK needs of all documentation users. Moreover, annotating researchers or architects may not know the interpretation of AK concepts and the AK needs of all documentation users by heart.

Questions can be used to classify design artifacts in the Zachman framework for enterprise architecture [121]. Typical questions gathered in our approach are mostly asked from the owner, designer, and builder perspective and from all abstractions (what, how, where, who, when, and why) described in the Zachman framework. Zachman argues from empirical observation that design artifacts (e.g., product descriptions and engineering documentation) can be classified by the users of these design artifacts. In our approach the users of AK evaluate classifications of AK concepts that are used to describe design artifacts.

In their TOVE enterprise modeling approach [43] Grüninger and Fox make use of ‘competency questions’ for ontology evaluation. Competency questions are considered ontology requirements and are used to evaluate an ontology based on its ability to answer the competency questions [43]. The notion of typical questions is similar to that of competency questions. Competency questions are however not refined and evaluated by users in order to construct an ontology (introduce ontological commitments), as is done in our approach with typical questions.

The On-To-Knowledge methodology [97] developed by Sure *et al.* uses competency questions to add relations to an ontology. In [110] Uschold and King however found competency questions to be too specific to guide early ontology development. Moreover, formalizing competency questions in first-order logic requires different skills from an ontology engineer than those required in the ‘typical question’ approach.

Acquisition of competency questions in these approaches is different to the acquisition of typical questions in our approach. For example, in the UPON methodology, proposed by Nicola *et al.* in [75], competency questions are gathered using interviews with domain experts, brainstorming and document analysis, whereas direct acquisition of typical questions from all users is proposed in our approach. Moreover, optimization of the efficiency and accuracy of AK needs acquisition and evaluation is proposed in our approach. This optimization is achieved by selecting different techniques for acquiring typical questions based on the commitment and accessibility of users.

The DILIGENT methodology, described by Pinto *et al.* in [79], focuses on distributed ontology development involving different stakeholders, possibly in separate locations, with varying needs and purposes. Ontology users can change an initial shared domain ontology in their local environment according to their needs. These users can then provide arguments for change requests to a central board which makes decisions and judgments on modeling of user needs and conflicting requirements. Similarly users give feedback on ontology concepts in our approach. Our approach however uses typical questions in the evaluation process.

CHAPTER 5. AN EXPLORATORY STUDY ON ONTOLOGY ENGINEERING FOR ARCHITECTURE DOCUMENTATION

Moreover, AK users in our approach not only evaluate AK concepts needed by themselves but also those needed by other AK users.

Kotis *et al.* describe their HCOME methodology in [59] in which users collaborate to solve conflicting interpretations of concepts in an ontology. HCOME allows ontology users to propose an updated ontology by themselves. This relies on different skills of ontology users as compared to our approach in which SA documentation users need the skill to sharply phrase their typical questions about AK.

5.7 Conclusions and Future Work

The use of ontology-based SA documentation can improve AK descriptions and retrieval. The implementation of ontology-based SA documentation requires acquisition and ontology modeling of the AK needed by SA documentation users. In software projects in industry it is important for organizations and individual users that this is done efficiently and accurately. For industry domains that are large, complex, and have many diverse users, this becomes challenging.

We devised a 'typical question' approach to ontology construction for SA documentation in the context of a large and complex software project. In this approach typical questions are used to acquire a tangible representation of AK needs from SA documentation users. AK concepts are identified from the typical questions and modeled in an ontology using coding techniques from Grounded Theory. Typical questions are also used for ontology evaluation and to identify conflicting interpretations of AK concepts between AK users working from different roles and disciplines.

We described contextual factors that influence the construction of ontologies for SA documentation in software projects, e.g., the specificness of the product domain and the accessibility of AK users. We found that the 'typical question' approach could be applied to build an ontology for many diverse AK users in the context of a large and complex software project. This is reasonable since: 1) the 'typical question' approach is based on the involvement of different roles 2) resolves conflicting interpretations between roles and 3) supports different forms of acquisition, e.g., interviews and emails, to handle different levels of commitment and availability. AK users evaluating the ontology all confirmed that the ontology was practical to work with and most AK users confirmed it was a correct representation of reality.

Our approach caters for changes in AK needs, however it cannot prevent ontology maintenance as AK users mostly provide AK needs for their current tasks. The

5.7. CONCLUSIONS AND FUTURE WORK

frequency of such maintenance depends on the frequency with which user tasks change.

These findings are based on a single case study of SA documentation for software-intensive products. Our exploratory case study was a first step to test if our 'typical question' approach is suitable for gathering enough explicit product-specific domain knowledge to produce an accurate AK ontology. We plan to do additional case studies, to generalizing our findings beyond one company and investigate if the 'typical question' approach can be applied by industry professionals. A comparative study of the use of other ontology engineering approaches, possibly combined with ours, to gather AK needed for both current and future tasks of AK users will be future work.

6

How Organisation of Architecture Documentation Influences Knowledge Retrieval

In this chapter we report case studies in two companies to investigate how the use of file-based and ontology-based documentation influences AK retrieval (RQ4). A common approach to SA documentation in industry projects is the use of file-based documents. This approach offers a single-dimensional arrangement of the AK. AK retrieval from file-based documentation is efficient if the organisation of knowledge supports the needs of the readers; otherwise it can be difficult or impossible to retrieve the knowledge. The ontology-based approach offers a multi-dimensional organisation of AK by means of a software ontology and semantic wiki, whereas file-based documentation typically uses hierarchical organisation by directory structure and table of content. We conducted case studies in two companies to study the efficiency and effectiveness of retrieving AK from the different organisations of AK. We found that the use of better AK organisation correlates with the efficiency and effectiveness of AK retrieval. Professionals who used the AK organisation found this beneficial.

In software industry, it is common practice to capture AK using file-based documents. This documentation approach has not changed for decades, however, it has various issues when retrieving AK:

- The AK that is searched for is often complex, covering different parts of a system and different stages of development. The AK needed to answer a question is often not found in one part of a document.
- The way architects organise the contents of a document is reflected in its table of contents, which provides an index on the AK. If a search for AK

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

is not supported by this table of contents, then the AK may not be easily found.

- If structuring of document content is not done properly, AK can become redundant and scattered across architecture views, sections, and documents. This is hard to prevent when there are many stakeholders with different AK needs.
- Cross-references between different sections and documents, e.g., in a traceability matrix, can help searching for scattered AK, however, they are hard to maintain when AK evolves.

These issues occur because file-based documents have a linear organisation of contents. This organisation limits the support for indexing contents. It results in documents that provide a single-dimensional arrangement of AK and that arrangement may not fit the needs of all AK users. Documentation that is not fitting for its users is not cost-effective [21] [31], yet documentation in industry is often 'one-size-fits-all' and does not serve specific users and their tasks well [80].

In Chapter 4 we described an ontology-based approach for SA documentation. In this chapter we report on two controlled industry experiments in which the ontology-based approach is compared to a file-based approach when software professionals retrieve documented AK. We measured the time-efficiency and effectiveness (in terms of precision and recall) of software professionals answering questions representative of their daily work.

Through the experiments we investigated how different organisations of AK in file-based and ontology-based documentation affect the efficiency and effectiveness of AK retrieval. Both documentation approaches are evaluated and compared.

We investigated the reasons why software professionals retrieved AK efficiently and effectively. We identified how the file-based and ontology-based AK organisation supported software professionals when they searched for AK. Explicit information in the AK organisations provided clues about the navigation path to relevant types of AK and relationships.

We quantified the usage of supporting AK organisation by analysing the individual search actions of the software professionals. We found that the use of AK organisation that supports a question has a medium to strong correlation with the efficiency and effectiveness of answering that question. Use of supporting AK organisation removed search uncertainty about the location and completeness of answers. This means that AK retrieval from SA documentation can be improved by providing more AK organisation that supports the questions of document users.

6.1. AK RETRIEVAL EFFICIENCY AND EFFECTIVENESS

Ontology-based documentation can improve AK retrieval by providing a more fitting AK organisation with more diverse possibilities to use the fitting AK organisation via multiple paths, however, it also incurs costs. These costs may not outweigh its benefits in certain projects. We estimated the costs, benefits, and return on investment of adopting ontology-based documentation in the two studied industry projects.

This work makes the following contributions:

- Compare AK retrieval from ontology-based and file-based AK organisation in two industrial experiments.
- Identify how and why AK organisation results in efficient and effective AK retrieval.

Section 6.1 details on the experiment setup and findings. Section 6.2 describes how AK retrieval is influenced by AK organisation. Section 6.3 reports a qualitative evaluation of the documentation approaches and experiment. Section 6.4 reports a cost-benefit analysis of adopting ontology-based documentation. Threats to validity are discussed in Section 6.5 and implications of this work are described in Section 6.6. Related work is discussed in Section 6.7. Section 6.8 reports our conclusions and future work.

6.1 AK Retrieval Efficiency and Effectiveness

6.1.1 Experiment Goal

We conjecture that the organisation of AK in file-based documentation causes certain issues with AK retrieval and that the AK organisation in an ontology-based documentation approach does not cause these issues. Given these two documentation approaches, we test their efficiency and effectiveness when retrieving AK. This allows us to investigate how the file-based and ontology-based AK organisation affects the efficiency and effectiveness of retrieving documented AK. The experimental goals are:

- **(A)** evaluate the AK retrieval **efficiency** of the file-based approach and the ontology-based approach to SA documentation.
- **(B)** evaluate the AK retrieval **effectiveness** of the file-based approach and the ontology-based approach to SA documentation.

The experiment was conducted in a software project at the R&D department of Océ technologies in the Netherlands and in a software project at LaiAn in China.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

Table 6.1: Variations between experiment domains

Items	Océ	LaiAn
Development process	Agile development in which business results delivery takes precedence over excessive documentation.	Waterfall development that stresses documentation in each development phase.
Scope of studied project and software documents	Software used in a series of document printing machines	Web-based information system for petition case administration of local government
Number of experiment document users	50~75	22~25
Language of experiment documents	English	Chinese
Number of experiment documents	8	1
Total size of experiment documents	79 pages, 3 diagrams, 1,794 paragraphs, and 3,183 lines, 13,962 words	46 pages, 20 diagrams, 348 paragraphs, 645 lines, and 8,538 words

We described the documents, development methodology, and recruitment of participants in the Océ experiment domain in more detail in Section 2.2.1. LaiAn is a software company that provides information system development and integration services for small and medium enterprises and local government. Table 6.1 details the variations between the experiment domains and the SA documentation used in the experiment.

A waterfall development approach is used at LaiAn, which requires the use of detailed upfront design documentation. Many parts of the information systems built at LaiAn are reusable in subsequent projects, and this reuse requires AK retrieval from SA documentation as well.

6.1.2 Experiment Participants

Table 2.1 in Section 2.2.1 gives the demographics of the experiment participants at Océ. Table 6.2 gives the demographics of the experiment participants at LaiAn. We asked technical employees that use the experiment documentation to participate and most agreed to this. At LaiAn, the project roles are roughly defined and assigned to individuals. A software engineer at LaiAn may also take on the role of deployment, test, and operations engineer; an architect may take

6.1. AK RETRIEVAL EFFICIENCY AND EFFECTIVENESS

on the role of requirements engineer and designer; and a project manager may take on the role of delivery and quality manager.

Table 6.2: Demographics of Participants at LaiAn

Number of participants	Primary role of participants	Average years in role at LaiAn	Average years in role	Average years working at LaiAn
15	Software engineer	1.87	5.47	1.87
5	Software architect	4.50	7.50	4.50
2	Software project manager	1.50	1.80	1.90

6.1.3 Experiment Materials

The materials used at both Océ and LaiAn consist, per experiment, of an SA documentation corpus, an ontology, and questions about the AK in the documentation corpus that are to be answered by experiment participants. Figure 6.1 gives an illustrated overview of how the experiment materials were constructed and used in the experiment.

File-based documents from Océ and LaiAn were used as input to construct ontology-based documentation. Software professionals at Océ provided examples of the types of AK concepts and relationships in the Océ domain, which were included in the Océ ontology. The experiment questions were constructed from documentation content by researchers. These questions were evaluated, and some rephrased or rejected, in a pilot study by software professionals. The software professionals in the pilot study did not participate in the experiment.

File-based Documentation

The documents used in the Océ experiment were previously described in Section 2.2.1. The LaiAn experiment uses one file-based document:

- One Software Architecture Document (SAD) of 46 pages. This single SAD contains all system goals, detailed requirements, system design, architecture design, design principles, subsystem, and components in a project at LaiAn. The document is mainly composed of text descriptions and UML activity and box-and-line diagrams. The document has an implicit view-based organisation. Views are not formally specified but the document sections and contents correspond to a logical, process, and use case view.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

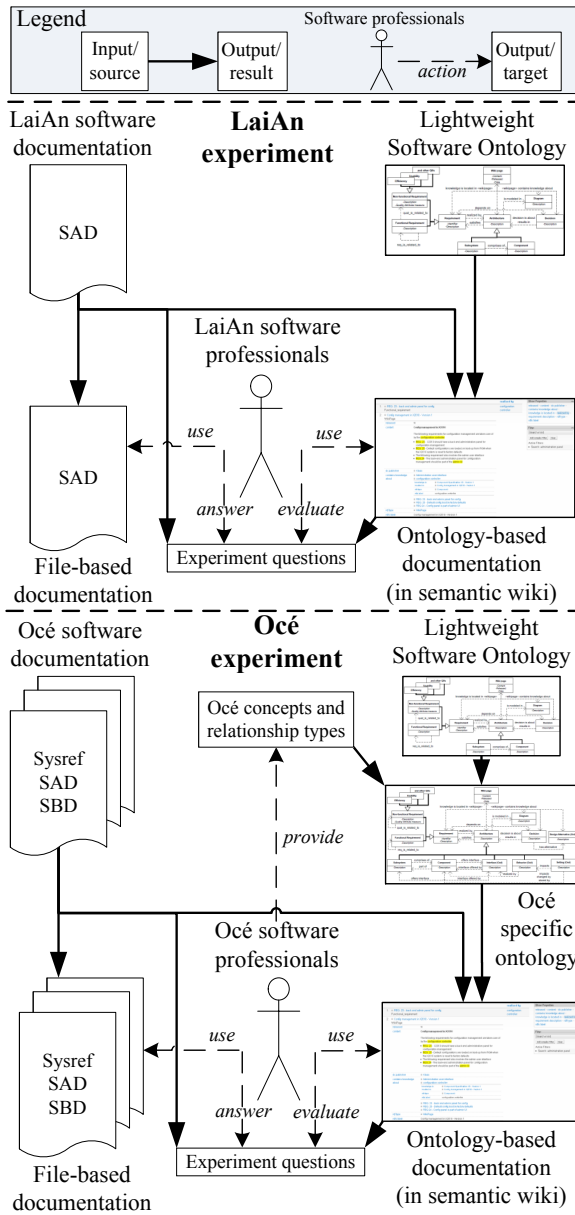


Figure 6.1: Construction and use of materials in experiments.

6.1. AK RETRIEVAL EFFICIENCY AND EFFECTIVENESS

All participants used Microsoft Word for reading and keyword searching in the file-based experiment documents. Océ participants used Windows file explorer for navigating directories, keyword searching across documentation, and opening documents in the experiment. In addition to the searchable text that specifies the architectural design, Océ participants used MagicDraw (a UML modelling tool) for viewing, tracing, and keyword searching in the architectural diagrams, whereas LaiAn professionals viewed the architectural diagrams as embedded static pictures in the file-based document. Use of the above tools is representative of the actual practice of Océ and LaiAn professionals.

Océ Ontology

The lightweight software ontology from [104] was directly used in the LaiAn experiment. We extended the lightweight software ontology to include Océ concepts and relationships types. We used the ontology engineering approach described in Chapter 5 for the ontology extension.

To do so, we asked Océ professionals to provide examples of the AK concepts and relationships they needed to retrieve from file-based architecture documentation in their daily work. Two examples of these AK needs are:

- "*What is the rationale behind this requirement? (And whom can we ask?)*"
- "*Which subsystem is responsible for fixing the XX defaults based on the device configuration?*"

We collected AK needs from 7 Océ professionals, among which software engineers, a software architect, and software project manager. These professionals work in multiple projects and printer product-lines, and the Océ ontology (see Figure 4.1) can be used as a general-purpose ontology in multiple projects.

From the information given by the Océ professionals we identified AK concepts and relationship types that were added to the lightweight ontology. We did not include identified AK concepts and relationship types that were not also recorded in the file-based documentation subset used in the Océ experiment. For example, AK concepts *'testcase'*, *'interface method'*, *'action'*, *'stakeholder'*, and their associated relationships, were not included. The accuracy and effort to construct the ontology are described in Chapter 5.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

Document Annotation

The Océ and LaiAn documents were entered in separate installations of ArchiMind. We identified and annotated 214 AK instances using the Océ ontology presented in Figure 4.1, namely;

- 27 wikipages and 3 diagrams
- 45 functional and 0 non-functional requirements¹
- 22 decisions and 3 alternative decisions
- 19 subsystems, 66 interfaces, and 15 components
- 8 settings and 6 behaviours

We identified and annotated 141 AK instances in the LaiAn documents using the lightweight ontology [104];

- 1 wikipage² and 20 diagrams
- 65 functional and 2 non-functional requirements. 1 system goal was annotated as a requirement.
- 7 decisions
- 21 components, 7 subsystems, and 18 architecture elements other than subsystems and components.

The annotations were verified by two software professionals at Océ and one software professional at LaiAn during a pilot study. We asked them whether specific AK instances were correctly classified (corresponding to an ontology class) and correctly interrelated by semantic relationships such as “*requirement X is realized by component Y*” and “*decision X is about subsystem Y*”.

After annotation, the ontology-based documentation contained the same AK as the file-based documentation. An ontology does provide extra information to organise AK. We want to test if this AK organisation helps professionals to retrieve AK.

¹Two non-functional requirements, performance and security, are explicitly and comprehensively described in the reference architecture documents, but not in the subset of these documents that was used in the Océ experiment. Other non-functional requirements are explicit in company-wide technical standards, and satisfied via the mechanisms, behaviour, and functional requirements specified in the reference architecture documents.

²Document content at LaiAn was stored as a property of the AK instances that this content described. This one wikipage provides an integrated overview of the AK instances using semantic annotations. As such, the use of wikipages in the LaiAn experiment is different from the Océ experiment.

Experiment Questions

Experiment questions were constructed at Océ and LaiAn from the content of experiment documentation. Researchers proposed experiment questions which were evaluated, and some rephrased or rejected, by two Océ professionals and one LaiAn professional in a pilot study. The experiment questions were constructed and evaluated in the pilot study based on four selection criteria that aim for a fair comparison between file-based documentation and ontology-based documentation. These selection criteria also ensure that we measure retrieval of documented AK, that is retrieval of AK which is explicitly present in the documentation, as opposed to retrieval of AK using memory, colleagues, specific expertise, other sources, or qualitative evaluation or understanding of AK. The selection criteria are:

- 1) The question is representative of the questions that documentation users ask during their job.

Criterion 1 is evaluated by the pilot participants to ascertain that the experiment questions are not 'artificial' and represent questions that professionals normally try to answer from documentation. Pilot study participants also ascertained that proposed questions were relevant for the tasks of Océ and LaiAn professionals in different software roles.

- 2) The answers can be found using the available AK and AK organisation in the documentation.

Criterion 2 is used to ensure that questions are supported by the available AK organisation in the experiment. For example, one of the selected questions is about settings and behaviour, which can be easily answered using the AK organisation in a file-based document about settings and less easily using another document about behaviour. In ontology-based documentation the classes '*Behaviour*' and '*Setting*' can be used to find answers, however, only one semantic relationship between these classes is defined, which makes it harder to find answers when starting to search from class '*Behaviour*'.

Criterion 2 is also used to ensure that the answers can be quantitatively assessed, i.e., that evaluators do not have to subjectively judge whether debatable answers to an open-ended question are either correct or incorrect. Because the information required to answer a question is available in documentation, the correctness of answers is not open for debate and subject to different interpretations. This criterion prevents that correct answers can only be found by participants with specific background knowledge that is not recorded in documentation. For example, answering a question about architectural trade-offs may require background knowledge that not all testers and software engineers have.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

Pilot study participants answered the proposed experiment questions and this ascertained that answers could be found using the available AK and AK organisation. The pilot participants also ascertained that answers were not open for debate or different interpretations, and could be quantitatively assessed.

- 3) The description of the AK that has to be found is consistent with similar descriptions of AK in the documentation.

Criterion 3 is used to ensure that the AK that has to be found does not have an atypical description and is recognizable for participants. Because the pilot participants had to find answers, they could ascertain whether the description of the answer was normal or atypical. For example, a pilot participant commented that it was normal that the answer to Océ question 1A has two descriptions in two documents.

- 4) The question has a similar interpretation between different participants.

We ascertained that software professionals had a similar interpretation of the proposed questions based on the comments, search actions, and answers of the pilot participants.

Based on the feedback of the pilot participants we replaced or rephrased the initially proposed experiment questions. For example, the proposed experiment question "*Based on which requirements has decision XX been made?*" has the following evaluation by a pilot participant: "*Answer [to this question] is not clear in documentation and open for discussion. Question is relevant though.*". We subsequently rejected this experiment question because it violated selection criteria 2, and we proposed a different question.

At Océ 13 experiment questions were proposed of which 6 were rejected and 3 were rephrased based on the evaluation by the two pilot participants. At LaiAn 8 experiment questions were proposed of which 4 questions were rejected in the pilot study. The following questions were used in the end:

Océ questions

Seven questions were accepted in the Océ experiment. The questions have been obfuscated for non-disclosure reasons: 'XX', 'YY', 'ZZ', and 'QQ' replace an actual software entity or concept.

1A: *Which settings have an impact on behaviour "XX"?*

1B: *Which settings have an impact on behaviour "YY"?*

- 2:** *Which requirements for behaviour "ZZ" should be satisfied (realized) by component "XX"?*

6.1. AK RETRIEVAL EFFICIENCY AND EFFECTIVENESS

3A: *Which decisions have been made about component “YY”?*

3B: *Which decisions have been made on the configuration of behaviour “XX”, “YY”, “ZZ”, and “QQ”?*

4A: *Which subsystem is interface “XX” part of?*

4B: *Which other interfaces are offered by this subsystem?*

LaiAn questions

For the LaiAn experiment we used 4 questions.

1: *Which requirements are realized by architecture design element “XX”?*

2: *Which requirements are related to requirement “YY”?*

3: *Which requirements does decision “ZZ” depend on?*

4: *Which architecture design elements are caused by decision “QQ”?*

“Architecture design element” refers to an implementable software artifact, e.g., a component or subsystem, in the LaiAn documentation.

The type of experiment questions proposed at LaiAn is similar to the type of questions at Océ to align the two experiments. The experiment questions involve relationships between AK and this is representative of the type of complex questions that Océ and LaiAn professionals ask in their daily job.

These types of questions are asked in multiple scenarios of SA documentation usage. For example, all questions can be asked during architecture refactoring and change impact analysis. Océ question 2 and LaiAn questions 1, 2, and 3 can be asked during architecture trade-off analysis and requirements verification.

6.1.4 Experiment Hypothesis

We formulate the following alternative hypotheses for experimental goal A and B presented in Section 6.1.1;

H_{1A} = *The use of the ontology-based approach for answering experiment questions results in better time-efficiency than the use of the file-based approach.*

H_{1B} = *The use of the ontology-based approach for answering experiment questions results in higher effectiveness than the use of the file-based approach.*

The null hypotheses state that there is no difference in efficiency and effectiveness between the approaches.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

Two independent variables (or ‘predictor variables’) are used in the experiment, namely the file-based and the ontology-based approach to SA documentation. Two dependent variables (or ‘response variables’) are used in the experiment. **Time** is used as a measure of efficiency. The harmonic mean of precision and recall, the **F1 score**, introduced by van Rijsbergen in [114], is used for measuring effectiveness:

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

where *recall* is the proportion of relevant items retrieved from the total set of relevant items in a system and *precision* is the proportion of retrieved items that is relevant in a result set. The relevancy of items, or ‘ground truth’, was verified with two Océ professionals and one LaiAn professional who did not participate in the experiment. *Recall* represents the completeness of AK retrieval and *precision* represents the correctness of AK retrieval. The use of precision and recall to measure search effectiveness is widely accepted in information retrieval research [88].

6.1.5 Experiment Procedure

We asked experiment participants to answer each of the questions using either the ontology-based approach or the file-based approach. We designed our experiment to be executed in two versions. Consecutive participants in the experiment were alternated between the two versions.

Both experiment versions 1 and 2 included an introduction and procedure (or ‘protocol’) at the start and a questionnaire at the end. Version 1 starts with an ArchiMind tutorial, questions 1 and 2 to be answered with the ontology-based approach, and questions 3 and 4 to be answered with the file-based approach. Version 2 starts with questions 1 and 2 to be answered with the file-based approach, the ArchiMind tutorial, and questions 3 and 4 to be answered with the ontology-based approach.

The experiment was designed to minimize biases when assigning participants to a treatment group and a control group. Each participant used both documentation approaches to retrieve AK and answer questions in the experiment. This design minimizes the chance that participants’ familiarity and preference for either approach could interfere with the results.

We chose to execute the experiment with each participant individually in a meeting room to avoid distraction for them and entropy in the experiment. We informed participants that their individual results were confidential to anyone other than the experiment supervisors.

6.1. AK RETRIEVAL EFFICIENCY AND EFFECTIVENESS

Océ participants were asked to think aloud, verbally state their answers, and their satisfaction with answers. LaiAn participants wrote down answers instead of verbalizing them. We asked all participants to stop searching when they were satisfied with the time spent on an answer and its perceived correctness and completeness. Participants were instructed that this satisfaction and the way they searched should reflect their daily practice.

6.1.6 Experiment Test Results

Using the Shapiro-Wilk and Kolmogorov-Smirnov tests, we found that the experiment measurements are not normally distributed. Therefore we applied the non-parametric Mann-Whitney-Wilcoxon test. Table 6.3 reports measurements and results³ of one-tailed tests.

Knowledge Retrieval Efficiency

The difference in time efficiency between the two approaches was statistically significant at the $p=0.05$ level for all Océ experiment questions, except for question 4A. This is shown in Table 6.3, in column ‘*p-value* test results’, for rows with ‘Seconds’ in column ‘measure’. Consequently, we reject the null hypothesis \mathbf{H}_{0A} and accept the alternative hypothesis \mathbf{H}_{1A} for all Océ questions except question 4A. Question 4A was very quickly answered with the file-based approach because AK about subsystems and interfaces is easily found in the AK organisation of multiple documents.

The difference in time efficiency between the two approaches was statistically significant for all LaiAn experiment questions, except for question 1, as shown in Table 6.3. Consequently, we reject the null hypothesis \mathbf{H}_{0A} and accept the alternative hypothesis \mathbf{H}_{1A} for all questions except question 1. The failure to reject the null hypothesis for LaiAn question 1 is explained by the short duration (5 minutes) of the ArchiMind tutorial given to participants. We observed that LaiAn participants took more time to use ArchiMind’s features during the first question compared to subsequent questions.

³The Mann-Whitney-Wilcoxon test was also applied on the same data in [30], however, the test in this paper was corrected for the many ties in *F1 score* measurements. Measurements for LaiAn question 1, answered by participant 1, were excluded as we unintentionally asked a slightly different question.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

Table 6.3: Time-Efficiency (Seconds), Effectiveness (F1 Score), and Statistical Test Results in Océ and LaiAn Experiment

Questions in Océ experiment	Measure	Average ontology-based	Average file-based	Difference	<i>p</i> -value test results	Effect size <i>r</i>
1A	Seconds	161	394	233	0.00914	0.46
	F1 score	0.97	0.96	0.01	0.28955	0.11
1B	Seconds	157	212	55	0.03232	0.36
	F1 score	0.85	0.65	0.20	0.02083	0.40
2	Seconds	229	382	153	0.00598	0.49
	F1 score	0.95	0.70	0.25	0.03672	0.35
3A	Seconds	148	401	253	0.00005	0.76
	F1 score	1.00	0.47	0.53	0.00050	0.65
3B	Seconds	197	374	178	0.00135	0.59
	F1 score	0.92	0.59	0.33	0.01694	0.42
4A	Seconds	73	78	5	0.44898	0.03
	F1 score	1.00	0.74	0.26	0.01673	0.42
4B	Seconds	40	64	24	0.01557	0.42
	F1 score	1.00	0.68	0.32	0.00762	0.48
All questions	Seconds	144	272	129	0.00001	0.85
	F1 score	0.96	0.68	0.27	0.00000	1.10
Questions in LaiAn experiment	Measure	Average ontology-based	Average file-based	Difference	<i>p</i> -value test results	Effect size
1	Seconds	251	259	7	0.29864	0.15
	F1 score	0.94	0.79	0.16	0.01245	0.49
2	Seconds	102	196	94	0.00214	0.61
	F1 score	0.91	0.43	0.48	0.00002	0.88
3	Seconds	216	263	47	0.03548	0.38
	F1 score	0.88	0.75	0.13	0.03108	0.40
4	Seconds	102	204	102	0.00193	0.62
	F1 score	1.000	0.98	0.02	0.15865	0.21
All questions	Seconds	168	230	62	0.00055	0.70
	F1 score	0.93	0.74	0.20	0.00000	0.95

Knowledge Retrieval Effectiveness

The difference in AK retrieval effectiveness between the two approaches was statistically significant for all Océ experiment questions, except for question 1A, as shown in Table 6.3. Consequently, we reject the null hypothesis H_{0B} and accept the alternative H_{1B} for all Océ questions, except question 1A.

The difference in effectiveness between the two approaches was statistically significant for all LaiAn experiment questions, except for question 4, as shown in Table 6.3. Consequently, we reject the null hypothesis H_{0B} and accept the alternative H_{1B} for all LaiAn questions except question 4. H_{1B} was not accepted for LaiAn question 4 and Océ question 1A because the file-based AK organisation provided much support for these questions (see end of Section 6.2.1 for more details).

6.2 How AK Organisation Affects AK Retrieval

The use of the ontology-based approach resulted in more efficient and effective AK retrieval than the use of the file-based approach in the experiment. The ontology-based AK organisation addresses the issues of file-based AK organisation, however, this does not explain how and why the organisation of AK affects AK retrieval efficiency and effectiveness, which is analysed in detail in this section. We need to analyse in detail how AK retrieval was influenced by the organisation of AK.

One of the objectives of this work is to identify how AK organisation may fit the AK retrieval needs of document users. We analyse how AK organisation supported participants in finding the types of AK and relationships between AK in each experiment question. We then compare how much of the file-based and ontology-based AK organisation gave support for the questions and we identify the usage of AK organisation by analyzing video recordings in the experiment. Next, we verify whether the use of supporting AK organisation results in efficient and effective AK retrieval. We then report how the AK organisation affected the search behaviour of participants.

6.2.1 AK Organisation

Fitting AK Organisation

The file-based documentation that was used in the experiments is organised by sections at LaiAn and by directories, documents, and sections at Océ. Ontology-

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

based documentation is organised by ontology classes and by semantic relationships between classes. Figures 6.2 and 6.3 depict the AK organisation that provides one or more paths to the answers for each experiment question, i.e., the directories, documents, sections, ontology classes, and semantic relationships that allowed participants to navigate towards answers or which contained answers.

Some of the nodes on a path to the answer explicitly relate to the question asked. For example, question 1A talks about settings and behaviour. The file-based documentation has a directory with software behaviour documents, which in turn has a document "behaviour print settings" with a section "system settings" and a subsection "settings" with text that makes it explicit where behaviour is described (see Figure 6.2). Here, the path to the answer has intermediate nodes which all fit the question. Or, in other words, the AK organisation fits the question. Conversely, when answering question 3A using the file-based documentation, the user has to go through various intermediate nodes that do not explicitly contain a reference to the question asked.

We term the nodes that explicitly refer to the question asked "fitting AK organisation". Shaded elements in Figure 6.2 and 6.3 denote fitting AK organisation.

"Fitting AK organisation" is identified as AK organisation that supports the questions in the experiment. We adopt the specific term "fitting AK organisation" and its definition because there may be other forms of support that an AK organisation provides for questions about AK. In the remainder of this chapter we use "fitting AK organisation" to refer to the supporting AK organisation that is identified and further investigated.

Influence of Fitting AK Organisation on AK Retrieval Efficiency and Effectiveness

The AK organisation was largely fitting for experiment questions in ontology-based documentation. The ontology-based AK organisation was overspecified for LaiAn question 3, which is about requirements whilst the ontology-based AK organisation provides a subdivision between functional and non-functional requirements. LaiAn question 4 is underspecified for the AK organisation in the ontology, as participants did not know exactly which architecture design elements had to be found.

The file-based AK organisation was only partially fitting for most the experiment questions. These questions were answered less efficiently and effectively in the file-based approach as compared to the ontology-based approach. Figure 6.2 and 6.3 show the average measured efficiency (in seconds) and effectiveness (in F1 score) per question as a means for comparison.

6.2. HOW AK ORGANISATION AFFECTS AK RETRIEVAL

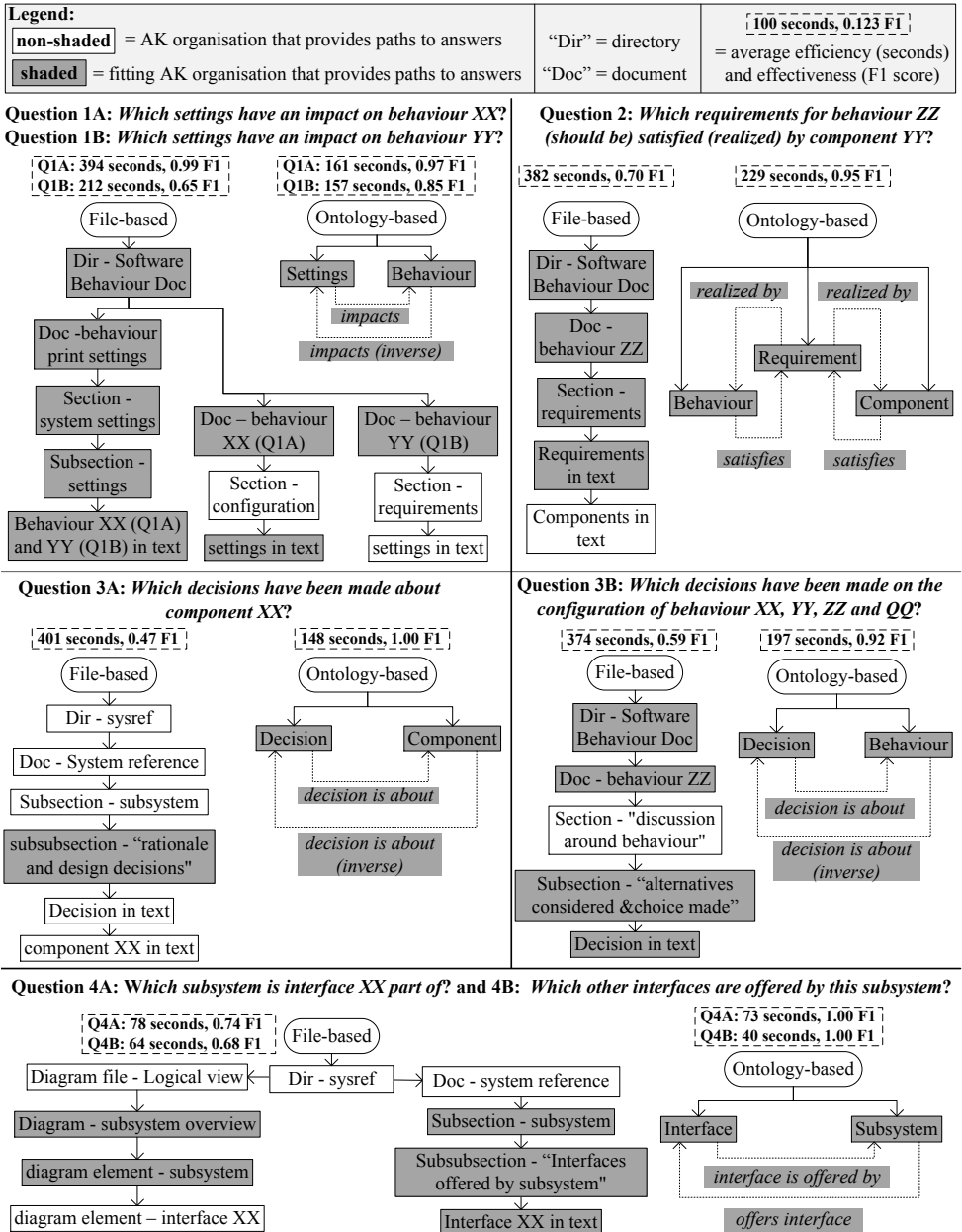


Figure 6.2: AK organisation for answering experiment questions at Océ

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

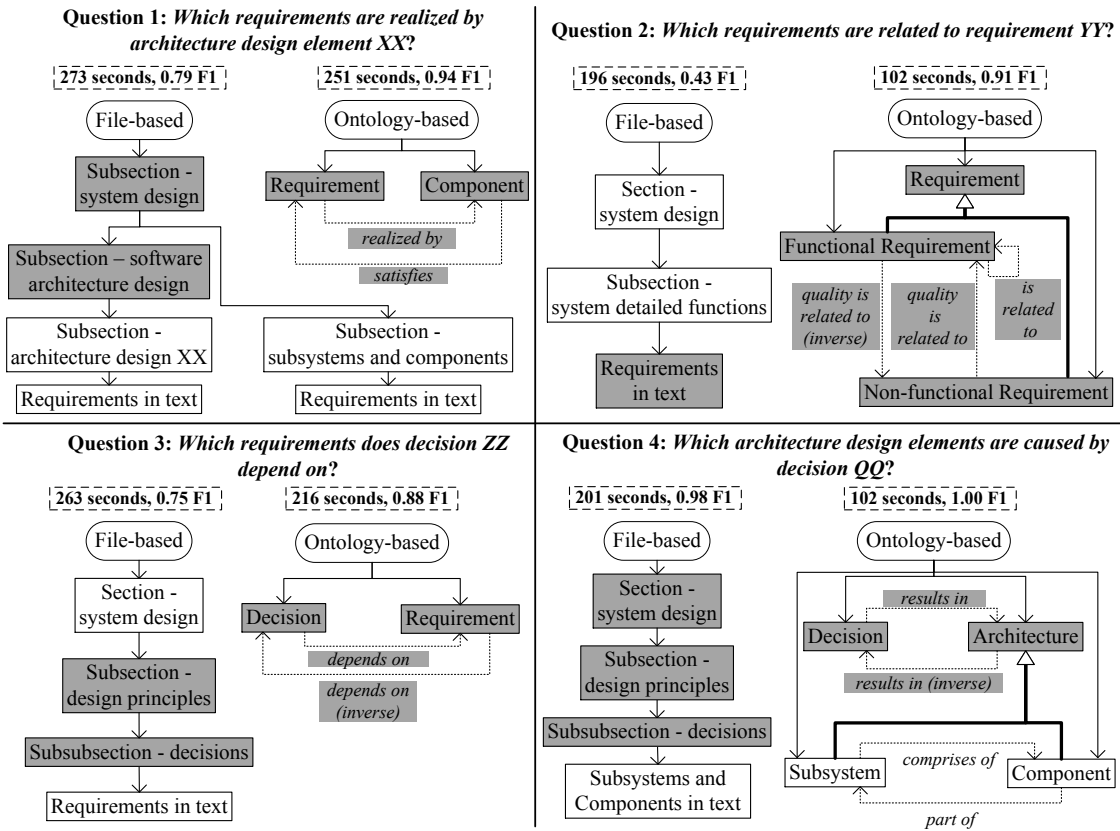


Figure 6.3: AK organisation for answering experiment questions at LaiAn

For example, the average time spent by participants answering Océ question 3A in file-based documentation was double that of participants using ontology-based documentation and they still retrieved less correct and complete answers. Similarly, the file-based AK organisation was not very fitting for LaiAn question 2, resulting in less efficient and effective AK retrieval as compared to ontology-based documentation.

Some questions are relatively well supported in file-based AK organisation, e.g., Océ question 1A and LaiAn question 4. The questions are often answered with similar efficiency and effectiveness in file-based and ontology-based documentation. This explains why there is no significant difference in effectiveness between the documentation approaches for these questions (also see Section 6.1.6).

6.2. HOW AK ORGANISATION AFFECTS AK RETRIEVAL

The file-based organisation however provided less paths to answers for Océ questions 2 and 3 and all LaiAn questions. Moreover, not all intermediate nodes on the path to answers were fitting. This provided less opportunity for participants to use fitting AK organisation when answering these questions. The semantic relationships in the ontology-based organisation allowed participants to find AK using multiple paths. For example, when answering Océ question 1A they could find answers by relating all settings to behaviour *XX* or vice versa, relating behaviour *XX* to all settings via semantic relationship '*impacts*'.

6.2.2 Use of Fitting AK Organisation

The analysis in Section 6.2.1 indicates that presence or absence of fitting AK organisation influences the efficiency and effectiveness of AK retrieval. However, this analysis does not tell us how participants used the available AK organisation.

There are answers that can be found in multiple file-based document sections, each with a varying amount of fitting AK organisation. Participants could use any of these sections to find answers. Moreover, participants could skip AK organisation by keyword searching on the names of AK instances. In this case they, e.g., skip the table of contents or class navigation and directly go to a document section or wikipage, respectively.

In order to analyze what AK organisation was used, we look at the search actions of participants during the experiment. We captured the search actions of participants by video recording what was shown on their monitor screen when they answered the experiment questions. Table 2.2 in Chapter 2 shows the different types of search actions in file-based documentation that we identified and encoded from the videos.

We measured use of AK organisation in about 6,000 search actions in over 11 hours of video recordings. We could not record videos of 9 out of the 22 LaiAn participants. Part of the video recordings of 2 participants in the Océ experiment were corrupted beyond repair.

"Use" of AK organisation might have different interpretations and meanings in different contexts. For example, participants might use an AK organisation by keeping in mind which document or section they are reading. This form of use is however hard to objectively measure. For our measurements, we consider fitting AK organisation to be used if 1) the fitting AK organisation appears on the screen of a participant, 2) the participant has enough time ⁴to recognize the fitting AK organisation, and 3) the participant follows the fitting AK organisation

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

and navigates to answers.

RatioTimeFitting is introduced as a metric which represents how much time participants spent using fitting AK organisation to answer an experiment question. *RatioTimeFitting* is calculated per participant per experiment question by dividing the '*time spent using fitting AK organisation*' by the '*total time spent searching for AK*'.

Figure 6.4 shows how *RatioTimeFitting* was calculated from the search actions of a participant answering Océ question 1A. The first two search actions involve use of a directory and document with titles that explicitly relate to a type of AK mentioned in question 1A, namely "behaviour". The 3 seconds spent on these actions is added to the '*time spent using fitting AK organisation*' and the '*total time spent searching for AK*'.

The next two search actions in Figure 6.4 do not involve use of fitting AK organisation; the participant quickly scrolled past the text in the title page and section 2. The 12 seconds spent is only added to the '*total time spent searching for AK*'. Actions 5 and 6 again involve use of fitting AK organisation because "setting" (an AK type in question 1A) is explicitly mentioned in text that is organised by a special layout, and because the text contains the answers to question 1A.

Participants using the ontology-based approach had a higher average *RatioTimeFitting* than participants using the file-based approach, i.e., they spent more time using fitting AK organisation during AK retrieval. On average the *RatioTimeFitting* of Océ participants was 0.72 when using the ontology-based approach and 0.39 when using the file-based approach. LaiAn participants had an average *RatioTimeFitting* of 0.70 when using the ontology-based approach and 0.63 when using the file-based approach.

The difference in *RatioTimeFitting* between the two approaches is smaller in the LaiAn experiment. This is due to the complexity of the file-based documentation: a single document was used at LaiAn whereas multiple documents and directories were used at Océ. Consequently, there was less non-fitting AK organisation to navigate in the less complex file-based AK organisation at LaiAn.

To verify that the use of fitting AK organisation influences AK retrieval, we test if a correlation exists between *RatioTimeFitting* and the efficiency and effectiveness of AK retrieval.

⁴Based on our observations from the experiment videos we chose to use 3 seconds as the minimum time required for participants to recognize and use fitting AK organisation in their searches. We observed that participants would act upon the fitting AK organisation after 3 seconds or more. After 3 seconds they would, e.g., open documents, click in ArchiMinds' class navigation, give answers from text containing fitting AK organisation, or talk about the AK organisation and surrounding AK.

6.2. HOW AK ORGANISATION AFFECTS AK RETRIEVAL

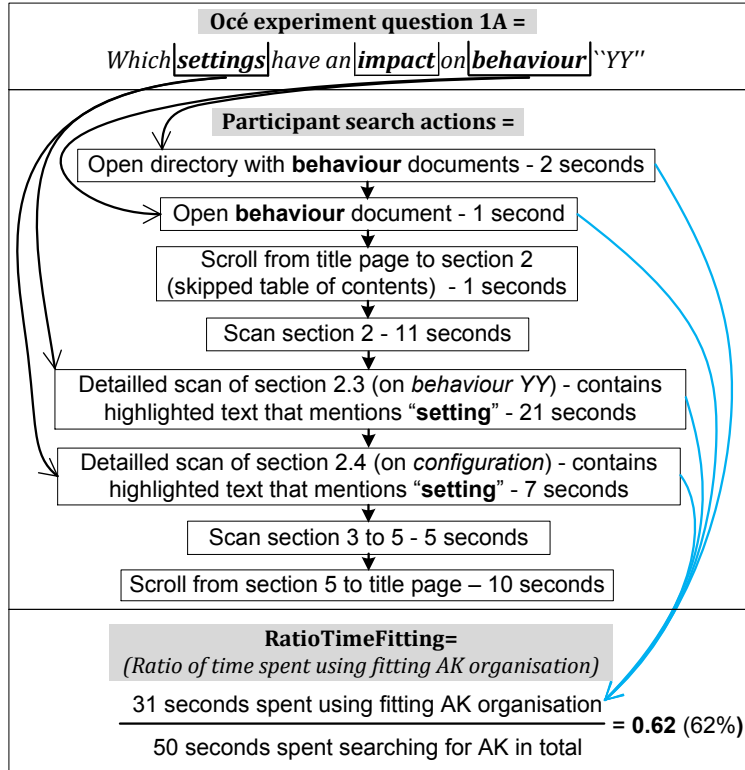


Figure 6.4: Example calculation of *RatioTimeFitting* from search actions of a participant using the file-based approach

We use the following hypothesis:

H_{1C} = *There is a correlation between the use of fitting AK organisation and the efficiency and effectiveness of AK retrieval.*

The null hypothesis states that there is no correlation.

Time-effectiveness is introduced to represent AK retrieval efficiency and effectiveness in a single metric which allows for testing of the hypothesis using two variables (*RatioTimeFitting* and *Time-effectiveness*). *Time-effectiveness* is calculated per answer in the experiment by dividing the *F1 score* (effectiveness) by the 'total time spent searching for AK' (efficiency).

A *Time-effectiveness* of 0.02 (e.g., for F1 score 1.0 divided by 50 seconds spent searching for AK) means that a participant is able to retrieve 2% of the complete and correct answer to an experiment question each second. Someone with a *Time-effectiveness* of 0.04 (or 4%) is twice as fast, e.g., by finding a complete and

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

correct answer in 25 seconds.

The *RatioTimeFitting* and *Time-effectiveness* for three Océ participants answering question 4 are not included in the test. These participants were not familiar with UML notations for interfaces and are not representative of the other 23 participants in this case.

Using the Shapiro-Wilk test, we found that the measurements for *RatioTimeFitting* and *Time-effectiveness* are not normally distributed. Therefore the non-parametric Spearman's rank correlation test is applied.

Application of Spearman's rank test indicates a strong correlation (coefficient $r = 0.67$) between *RatioTimeFitting* and *Time-effectiveness* in the Océ experiment and a moderate correlation (coefficient $r = 0.48$) at LaiAn. These test results are statistically significant at the $p=0.01$ level with a (2-sided) P-value of 6.98^{-24} for the Océ measurements and 0.00035 for the LaiAn measurements. Consequently, we reject the null hypothesis H_{0C} and accept the alternative hypothesis H_{1C} .

Figure 6.5 depicts the correlation in a scatterplot, where the x-axis displays *RatioTimeFitting* and the y-axis displays *Time-effectiveness*. Each dot in the scatterplot represents a single participant answering a single question. Dots in the upper right corner represent participants that took relatively little time to find a correct and complete answer (i.e. high *Time-effectiveness*) whilst primarily using fitting AK organisation. Dots in the lower left corner represent participants who did not find complete and correct answers quickly whilst using little fitting AK organisation. Figure 6.5 shows that increased use of fitting AK organisation (*RatioTimeFitting* on the x-axis) leads to increased *Time-effectiveness* of AK retrieval (on the y-axis).

6.2.3 AK Organisation and Search Behaviour

The test for correlation and our observations in the experiment videos give evidence that when participants used fitting AK organisation they:

1. quickly recognized the types of AK and relationships between AK.
2. quickly and correctly recognized the location of answers.
3. less often misinterpreted the descriptions of AK because the type of AK and the context of AK (i.e. relationship to other AK) was explicit.
4. continued searching if they had not retrieved all correct answers. They had a better understanding where to find different types of AK and relationships between AK.

6.2. HOW AK ORGANISATION AFFECTS AK RETRIEVAL

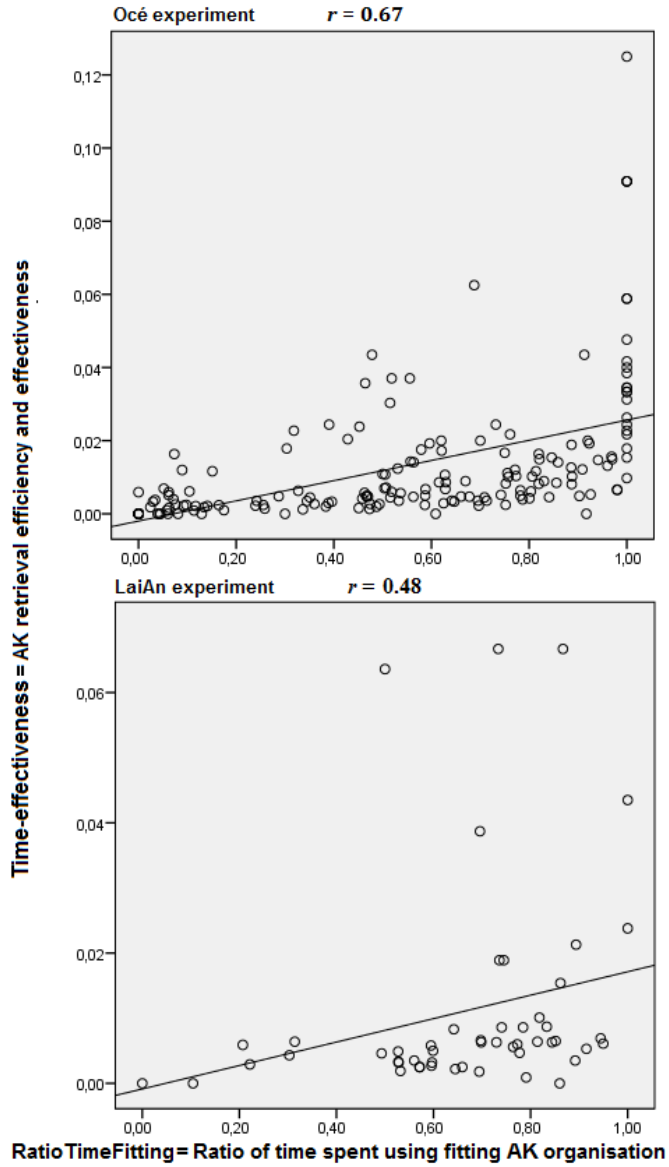


Figure 6.5: Correlation between *RatioTimeFitting* and *Time-effectiveness* when answering Océ and LaiAn experiment questions in the file-based and ontology-based documentation approach

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

When participants used less fitting AK organisation they often gave less correct answers. In this situation participant also spent more time searching documentation to verify that they found a complete (recall) and correct (precision) answer.

Participants navigated many file-based documents and sections when they encountered little fitting AK organisation. For instance, we measured from the video recordings that participants answering Océ question 3A navigated 3.6 directories and 4.7 file-based documents on average. In the ontology-based approach they however navigated only 1.8 classes and 1.2 semantic relationships on average.

In Chapter 2 we report an in-depth analysis of participants' search behaviour when they used the file-based documentation approach in the Océ experiment. We found that the participants had to deal with search uncertainty when fitting AK organization was missing in part of the navigation path to the answers. As a result, participants were not always certain in which document or section AK was located. Keyword searching helped to locate AK, but often took much time and gave incomplete results due to spelling variations, abbreviations, and synonyms. Participants said they were uncertain about the completeness and correctness of 38% of their answers from file-based documentation.

Participants could use the ontology-based AK organization by listing class instances in an overview and by faceting and filtering AK via semantic relationships. During the experiment we observed that these search features allowed participants to quickly check the completeness of their answers and remove search uncertainty. This use of ontology-based AK organization prevented errors and wasted time that might otherwise occur when dealing with search uncertainty in file-based documentation.

However, participants had to learn how to use the AK organisation in ArchiMind, reducing its positive effect on efficiency and effectiveness. For example, some participants were uncertain about how to use ontology-based AK organisation to filter AK in ArchiMind.

Several participants commented in a questionnaire (reported in Section 6.3) that the semantic relationships between AK and the class instance overview in ontology-based documentation allowed them to check for completeness and provided determinism to their answers. Participants commented that the use of semantic relationships for AK structuring, traceability, and navigating was helpful, and several commented that the semantic relationships removed search uncertainty and gave more search options to find relevant AK. When asked about issues with searching in file-based documentation, participants commented that indeterminism and difficulties in ensuring completeness of answers are problematic, which corresponds with findings about search uncertainty in Chapter 2.

6.3 Qualitative Evaluation

After the experiments we asked each participant to fill in a questionnaire, reported in Table Table 6.4, in which the file-based and ontology-based approach (referred to as 'ArchiMind') are evaluated. Table 6.5 reports an evaluation of the ontology and experiment by a subset of the Océ participants during a workshop and by LaiAn participants during meetings after the experiments took place.

The Océ workshop was announced via email and posters to invite software professionals working in the location where the experiment took place. The evaluation form (the basis for Table 6.5) could be collected when leaving the room where the workshop took place. We emailed LaiAn experiment participants to ask for their interest in evaluating the experiment findings. 10 LaiAn participants indicated that they were willing to fill in the evaluation form and we selected 6 participants by considering a wide coverage and even distribution of their roles.

6.3.1 Evaluation of Documentation Approaches

Most participants evaluate the ontology-based approach and its search mechanisms as being better than the file-based approach for searching AK. Participants generally feel it is worthwhile to implement ontology-based documentation in their company as long as its benefits outweigh the costs and if enough support is given.

Table 6.4: Questionnaire about File-based and Ontology-based Approach

1: <i>When searching for software knowledge, would you evaluate ArchiMind, compared to normal documentation, as:</i>
Océ: Better: 24 (92.3%) Worse: 0 (0%) Making no difference: 2 (7.7%)
Most participants comment that they find (semantic) relationships important and useful when searching software knowledge. The search mechanisms, facets, structure, and (centralized) accessibility are also found useful by participants.
LaiAn: Better: 14 (63.6%) Worse: 6 (27.3%) Making no difference: 0 (0%) No opinion: 2 (9.1%)
Some participants evaluate ArchiMind as worse because they feel that the UI is unsuitable for software documentation. This may be partially due to the lack of in-wikipedia annotation in the LaiAn experiment (also see Section 6.1.3).
<i>continued on next page...</i>

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

2: *Do you think that ArchiMind can provide you with better search mechanisms than currently at your disposal?*

Océ: Yes: 25 (96.2%) No: 0 (0%) I do not know: 0 (0%) No opinion on this: 1 (3.8%)

Most participants comment that the semantic relationships are useful for searching.

LaiAn: Yes: 13 (59.1%) No: 2 (9.1%) Some better, some not: 7 (31.8%) No opinion: 0 (0%)

Most participants find that ArchiMind provides meaningful traceability information. Some participants feel that the search mechanisms are not very convenient in some situations. For example, different levels of requirements exist, from system goals to detailed requirements, and users cannot distinguish between these levels when searching requirements.

3: *Do you think it is worthwhile to set up a semantic wiki at your company for searching software knowledge & documentation management?*

Océ: Yes: 17.5 (67.3%) No: 2 (7.7%) I do not know: 6.5 (25%) No opinion: 0 (0%)

Most participants comment they do not know whether the benefits of the ontology-based approach outweighs the costs. Other elaborations given are that enough effort should be invested, authorization should not be an obstacle, training should be provided and that the knowledge in the system should be complete, maintained well and reviewed by an expert. One participant chose both options 'yes' (for management) and 'I do not know' (for searching).

LaiAn: Yes: 14 (63.6%) No: 3 (13.6%) I do not know: 4 (18.2%) No opinion: 1 (4.5%)

Most participants support the idea that it is worthwhile to set up a semantic wiki within their companies. Some participants tend not to change to semantic wiki when current documentation tools work well. Other participants are concerned about the conformance issue of documentation (e.g., document template and structure prescribed by customers), especially in outsourcing projects. Two participant chose both options 'Yes' and 'No' with their arguments. For example, one thinks that the answer of this question depends on the size of the project: traditional documentation tools, like Office Word, are appropriate for small projects and the semantic wiki is better for large and complex projects.

continued on next page. . .

6.3. QUALITATIVE EVALUATION

4: *Do you experience troubles in your daily job when searching for software knowledge using the standard documents?*

Océ: Yes: 23 (88.5%) No: 3 (11.5%)

Most participants comment that documentation is often outdated. Other elaborations are that documentation is incomplete, indeterministic, difficult to access or even hidden, contained in (too) many (scattered) sources, hard to verify whether trustworthy, costly to keep up to date, lacks detailed information, and has conflicting requirements.⁵

LaiAn: Yes: 8 (72.7%) No: 3 (27.3%) (this question was answered by half of the participants)

Most participants comment that only few documents are really useful and have been used in the software development because it is difficult to find the information they want. Other issues are lack of traceability in requirements and design specifications, and difficulty in performing impact analysis using documents.

5: *From which sources do you normally get knowledge about the software made at your company?*

Océ: Most often mentioned are colleagues, then documents, source code, Sharepoint, Docfinder, and CMSynergy.

LaiAn: Documents are most often mentioned and after that colleagues and source code.

6: *From which types of documents do you normally get knowledge about the software made at your company?*

Océ: Most participants use SBDs. SADs, interface and functional specifications, diagrams, technical reports and source code are also used as well as impact analysis, high level architecture, Sysref, and module design documents.

LaiAn: Most participants use requirement and architecture documents. Design, bidding, business process, project planning, test, traceability, and API documents are also used as well as source code and customer surveys.

7: *What percentage of your time do you daily spend on searching and retrieving software knowledge?*

Océ: 19.75%. The answers range from 0% to 50% or more. This question was answered by half of the participants.

LaiAn: 29.17%. The answers range from 10% to 60%. This question was answered by all the participants.

⁵Océ successfully applies an agile development methodology to encourage creativity and productivity. The drive to deliver business results is strong, and this takes precedence over writing excessive documentation.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

Table 6.5: Experiment and Ontology Evaluation at Océ and LaiAn

1: <i>Do you believe in the experiment results? (do the results represent reality or are they artificial)</i>
Océ: Yes: 4 To a certain extent: 1 No : 0 LaiAn: Yes: 5 To a certain extent: 1 No : 0
2: <i>Are the experiment results limited to the specific question set used in the experiment?</i>
Océ: Yes: 1 Maybe: 2 No: 2 LaiAn: Yes: 2 Maybe: 2 No: 2 Océ respondents comment that there are many questions and that it will be hard to model the entire working field.
3: <i>Are the experiment questions relevant to your job and representative of the questions you ask during your job?</i>
The Océ and LaiAn respondents evaluate all questions as relevant and representative for their jobs except for Océ question 3A (decisions made about a component) and LaiAn question 1 (requirements realized by architecture design) which one Océ and one LaiAn respondent evaluate as irrelevant and not representative.
4: <i>Is the ontology model used in the experiment a correct representation of reality?</i>
Océ: Yes: 3 To a certain extent: 2 No: 0 LaiAn: Yes: 4 To a certain extent: 2 No: 0
5: <i>Should there be more or less concepts in the model?</i>
Océ: More: 4 The same amount: 1 Less : 0 LaiAn: More: 2 The same amount: 4 Less : 0 An Océ respondent indicates that more specific domain knowledge should be added.
6: <i>Is it practical to work with the predefined model of software (architecture) knowledge?</i>
Océ: Yes: 5 To a certain extent: 0 No : 0 LaiAn: Yes: 5 To a certain extent: 1 No : 0
7: <i>Does the model help in reasoning about what knowledge is in the documents and what should be in documents?</i>
Océ: Yes: 5 To a certain extent: 0 No : 0 LaiAn: Yes: 6 To a certain extent: 0 No : 0

6.3.2 Evaluation of Experiment and Ontology

Table 6.5 shows that most respondents consider the experiment questions to be relevant for their job and representative of the questions they ask in their daily work. This evaluation by experienced documentation users indicates that we asked the right questions in both experiments.

Part of the respondents think that the experiment results are limited to the specific question sets. Their remarks suggest that it will be challenging to give ontology support for all domain knowledge and questions asked. Even though the respondents generally evaluate the ontology as realistic, most Océ respondents think there should be more domain concepts in the ontology. This may very well reflect the specific domain in which they work.

6.4 Cost-Benefit Analysis

The experiment results show that ontology-based documentation can provide benefits by improving the efficiency and effectiveness of AK retrieval. However, there are also costs associated with setting up ontology-based documentation. A concern that Océ and LaiAn practitioners have with adopting the ontology-based approach in their projects is whether its benefits outweigh its costs (see question 3 in Table 6.4). In this section we provide a cost-benefit analysis of using ontology-based documentation in the studied projects at Océ and LaiAn.

Costs and benefits are undeniable factors when discussing the documentation of SA [10]. A recent systematic literature mapping in [122] however shows that there is very little work about the cost aspect of software documentation. Even less work is published that quantifies both the costs and benefits, or the return on investment of using software documentation. One notable example is the work by Garousi *et al.* in [38], where a cost-effectiveness index of technical software documents is calculated by dividing the number of times that a document is accessed or downloaded (benefit) by the time spent editing this document (cost).

Similarly, we use measures that represent costs and benefits of ontology-based documentation in order to estimate its return on investment when it replaces file-based documentation. We estimate costs from the recorded time spent on creating the ontology-based documentation in the experiments. We estimate benefits from the efficiency and effectiveness measurements in the experiment and document usage estimates by professionals. The estimates by professionals are crude, e.g., "*I estimate 20% to 25%*", and therefore the cost-benefit analysis is indicative.

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

6.4.1 Costs and Benefits in Océ Project

We spent 4 hours installing and configuring ArchiMind. Around 40 hours were spent to build the Océ ontology and semantically annotate AK in the 79 pages in the Océ experiment document subset. We estimated that the total set of actively used product-line reference architecture documents is 2024 pages in size. Building an ontology for, and semantically annotating the AK in the total active documentation set is estimated to cost around 1028 hours.

Océ participants estimate that on average they spend 19.75%, or 1.6 hours, of their daily working time on retrieving software knowledge (see question 7 in Table 6.4). By consulting 13 Océ professionals in most project roles we estimated that out of the 1.6 hours each day, 16.6 minutes is used on average to retrieve AK from the product-line reference architecture documents.

At least 50 Océ professionals use the reference architecture documents, as shown in Table 6.1. The Océ experiment participants spent 47.06% less time on average when they used the ontology-based approach, compared to the file-based approach. From this we estimate that on average 6.5 hours can be saved each workday by the 50 Océ professionals combined.

6.4.2 Cost and Benefit in LaiAn Project

We spent 4 hours installing and configuring ArchiMind at LaiAn. Around 24 hours were spent to semantically annotate the single SA document of 46 pages which is the total document set that specifies the SA in the studied project.

LaiAn participants estimate that on average they spend 29.17%, or 2.3 hours, of their daily working time on retrieving software knowledge (see question 7 in Table 6.4). By consulting a subset of the participants that cover all project roles we estimated that out of the 2.3 hours each day, 17.4 minutes is used on average to retrieve AK from SA documentation.

At least 22 LaiAn professionals use the experiment document, as shown in Table 6.1. The LaiAn experiment participants spent 26.96% less time on average when they used the ontology-based approach, compared to the file-based approach. From this we estimate that 1.7 hours can be saved each workday by the 22 LaiAn professionals combined.

6.4.3 Return on Investment

We give an estimation of the period after which the replacement of file-based documentation with ontology-based documentation in the studied projects results in an overall time gain. This is a coarse estimation because it depends on many variables; the return on investment may be sooner or later depending on, e.g., the learning curve of professionals using the semantic wiki, proficiency of professionals that semantically annotate the documents, and the exact usage of SA documentation across different software project phases.

We have not measured the cost of maintaining ontology-based SA documentation, e.g., updating document content in wikipages and updating AK organisation in the ontology to support evolving AK needs.

We estimated in Section 6.4.1 that the ontology-based approach can save 6.5 hours each working day on average at Océ, and this accumulates to 1690 hours time savings each year. This indicates that the return on the 1028 hours investment cost, i.e., the break-even point, is around 7 months (or 158 working days) after ontology-based documentation is introduced. The time savings after the 7 months apply to product-line reference architecture documentation that is used for multiple years.

We estimated that the ontology-based approach can save 1.7 hours each working day on average at LaiAn. This indicates that the return on the 28 hours of investment is around 3 weeks (or 16 working days) after ontology-based documentation is introduced. The time savings after the 3 weeks apply to SA documentation that is used for several months in the studied project.

The ontology-based approach also improves the *effectiveness* of AK retrieval. Having more correct and complete information prevents errors, which in turn also saves time that would normally be spent on correcting these errors.

6.5 Threats to Validity

In our experiment plan, we accounted for possible threats during the experiment design. We followed guidelines from [54] for reporting the experiments and its threats to validity.

6.5.1 Construct Validity

The researchers who created the ontology later also proposed the experiment questions. The creation of the ontology and the preparation of the experiment questions were done as separate activities at different times. The questions were not set based on the ontological structure. The experiment questions were checked by professionals to ensure that they are representative of questions asked by professionals in the studied domains. To ensure that the answers could be found in both approaches, the researchers checked if the answers could be found using both the ontology-based and file-based AK organisation. We mitigated this potential bias by using experiment questions based on the experiment documentation content and evaluation of four selection criteria by professionals in a pilot study (see Section 6.1.3). The group of pilot study participants was different and independent from the software professionals that provided Océ concepts and relationship types for Océ ontology extension.

The use of ontology-based documentation did not always favour AK retrieval efficiency and effectiveness. For instance, several experiment questions could be answered using the inverse of semantic relationships in the ontology, namely, Océ questions 1A, 1B, 3A, and 3B and LaiAn questions 2, 3, and 4. This negatively impacted the performance of the ontology-based approach in our experiment.

6.5.2 Internal Validity

A subset of the documentation from one particular Océ project was used. Participants from that project had, more than other Océ participants, prior knowledge of this documentation. However, the document and question types used in the experiment were generic. To avoid prior knowledge that could bias our results, participants were not informed about how much and precisely which documents were present in the documentation subset used in the experiment. Moreover, they were instructed to always find and verify possible answers in the experiment documentation despite any prior knowledge.

In the Océ experiment the description of interfaces, required for question 4B, was outdated in the system reference document. Océ professionals verified that presence of outdated documents is a common situation, e.g., during development. We chose to not update any AK in order for the file-based and ontology-based document content to be identical and realistic. This situation does not affect the interpretation of our results.

A possible bias is that participants that evaluated the experiment results (reported in Table 6.5) are more curious and receptive to new ideas and tools, and

this is a potential threat to validity. We tried to mitigate this threat by sending an open invitation for the Océ workshop and by selecting a subset of LaiAn participants based on a wide coverage of their roles (see Section 6.3).

6.5.3 External Validity

Even though exact replication between experiments in software engineering is unattainable [13], we aimed for maximum consistency between the experiment procedures in the Océ and LaiAn experiments. Several variations between the experiment domains, e.g., those reported in Table 6.1, indicate that the results are generalizable to other software project domains. The use of SADs, SBDs, system reference-, and design documents can be considered generic documentation practice in industry.

The specific set of questions asked in the experiment limits generalization, even though we verified that the questions are representative of the questions that the industry professionals ask in their daily work. For example, the experiment questions involve traceability between AK, which may not be fully representative of questions in other software projects. Moreover, the questions involve retrieval of AK that is explicitly present in documentation, as opposed to, e.g., retrieval or evaluation of AK from memory and colleagues.

We did not investigate how *frequent* the experiment questions are normally asked in the studied projects. This limits generalization because certain questions may be asked more frequently than others, and thus have a greater impact on AK retrieval efficiency and effectiveness.

Architectural models were searched and traced via a UML modelling tool (Magic-Draw) in the file-based approach at Océ, and viewed as static pictures embedded in the file-based document at LaiAn. The use of architectural modelling tools in our experiment may be different from other software industry projects, which may adopt advanced architectural modelling tools with extensive search, AK annotation, cross-referencing, and tracing features. A file-based documentation approach that includes the use of more advanced architectural modelling tools may provide better AK retrieval efficiency and effectiveness than the file-based approach studied in our experiment.

Several factors in the cost-benefit analysis limit generalization. The semantic annotations were manually applied, which is more costly than the use of semi-automatic annotation. The researchers had little domain knowledge, which increased the time required for Océ ontology extension. The studied software projects at Océ and LaiAn are architecture-driven, which increases the usage of

CHAPTER 6. HOW ORGANISATION OF ARCHITECTURE DOCUMENTATION INFLUENCES KNOWLEDGE RETRIEVAL

SA documentation and in turn increases the potential benefits ontology-based SA documentation.

6.5.4 Conclusion Validity

As the data collected in the experiment is not normally distributed, we cannot calculate statistical power under the assumption of normal distribution. The increased chance of a Type II error (false negative), when using a non-parametric test on normally distributed data, does not apply here [37].

6.6 Implications

6.6.1 Implications for Practitioners

We found that the use of AK organization that is fitting for questions correlates with the efficiency and effectiveness of answering these questions. Use of fitting AK organisation helped participants to quickly identify the location of answers and correctly recognize the answers. Lack of fitting AK organisation introduces search uncertainty about the location and completeness of answers, which caused participants to waste time searching for AK in wrong locations and miss answers.

This means that AK retrieval from existing file-based documentation in industry can be improved by providing more fitting AK organisation for the questions of its users. This can be done by first identifying what questions document users ask and then creating an AK organisation that explicitly denotes where the types of AK and relationships between AK in these questions can be found.

If there are many users in a project, with many different questions about inter-related AK, then an ontology-based approach may be more cost-effective than a file-based approach. An ontology is non-linear and can provide a fitting AK organization for many questions about interrelated AK without introducing redundancy and scattered AK descriptions. When there are many AK users this results in large benefits that may outweigh the costs of creating ontology-based documentation.

However, a linear file-based AK organisation can be fitting for the set of questions that is most frequently asked. Our findings in Section 6.2 suggest that the AK retrieval efficiency and effectiveness of file-based and ontology-based documentation is similar when both approaches provide fitting AK organisation for questions. A file-based approach could be more cost-effective, e.g., in small projects with few

AK users, because creation of a file-based document for a few questions that are frequently asked could be less costly than setting up ontology-based documentation, whilst both approaches provide similar benefits in this case.

We found that a tutorial on the features of the ArchiMind semantic wiki tool is helpful to efficiently and effectively use ArchiMind for the first time. The participants in the Océ experiment followed a 30 to 45 minute tutorial on ArchiMind and used the ontology-based approach more efficiently and effectively at the start of the experiment as compared to the LaiAn participants who followed a 5 minute tutorial.

6.6.2 Implications for Researchers

Software and its architecture is decomposed by separation of concerns and this is reflected in its documentation. The linear nature of file-based documentation imposes practical limitations when comprehensively describing the relationships between AK that are relevant for the concerns of document users. The limitations of the linear file-based documentation format are overcome in ontology-based documentation.

In our experiment, we demonstrated that an ontology-based organisation can support many relationships between AK and prevent redundancy in the descriptions of AK. File-based documentation supported less relationships between AK and did contain redundant and scattered AK descriptions.

In our analysis we quantified the impact of AK organisation on the efficiency and effectiveness of AK retrieval. The correlation between the use of fitting AK organisation and the time-effectiveness of AK retrieval explains why ontology-based documentation was more effective and efficient than file-based documentation. Moreover, the analysis of the AK organisation and the correlation apply to both documentation approaches and show that one can improve AK retrieval from SA documentation in general by providing more fitting AK organisation.

Viewpoints and views are used to frame and address the concerns of stakeholders [2]. Our findings suggest that stakeholders retrieve AK more efficiently and effectively when view-based descriptions have a fitting AK organisation for the questions that follow from their concerns. Combining view-based architecture descriptions with an ontology-based approach, as proposed by Tamburri in [99], seems promising in this light.

6.7 Related Work

López *et al.* [68] proposed the Toeska Rationale Extraction (TReX) approach to recover, represent, and explore design rationale in text documents, including in-page semantic annotations. The Toeska ontology, based on ArchVoc [8], is used together with the NDR ontology. Semantically annotating AK concepts outside these ontologies requires adaptation of the TReX tools, ontologies, and experience with natural language processing, whereas our approach only requires ontology adaptations. A web-based rationale browser allows for retrieval of AK using faceting and linking to document sources, and these features are also provided in ArchiMind. It is not specified in [68] whether this rationale browser supports the same search features as ArchiMind, e.g., keyword search, ontology browsing, and filters. Evaluation by comparing the precision, recall, and time-cost of AK retrieval between a file-based and ontology-based approach is similar to our experiment setup. However, complete recovery of all AK in a document set by students and by an automated rationale recovery system is tested in [68], whereas we tested AK retrieval using a specific set of questions that was answered by industry professionals. Their case study provides evidence that AK recovery is more effective when using an ontology-based approach.

Jansen *et al.* [51] proposed a method, and the associated Knowledge Architect tool suite, for semantic annotation of AK in SA documents. An ontology is constructed by identifying AK concepts in SA documentation, whereas we use a general-purpose AK ontology and AK needs collected from software professionals for Océ ontology extensions. Their approach uses an MS word plug-in for semantically annotating and viewing AK ontology instances and a tool for navigating AK instances and their relationships. Different tools are used for semantically annotating, viewing, and navigating AK, whereas ArchiMind provides a single integrated interface for this. In [51] evidence is provided that the use of an ontology-based approach improves the understanding of AK in terms of improved efficiency and quality of comments during architectural review. This is different from our evaluation, which is in terms of AK retrieval efficiency and effectiveness.

6.8 Conclusions

The major contribution of this work is the empirical evidence that explains how the organisation of documented AK impacts the efficiency and effectiveness of retrieving that AK. We conducted experiments in which software professionals

answered questions about AK that are representative of questions they ask in their daily work. Part of the available AK organisation was fitting for AK retrieval; it explicitly denoted the AK types and relationships between AK specified in the experiment questions. We found that the usage of fitting AK organisation correlates with the efficiency and effectiveness of AK retrieval.

We quantified and verified the impact of AK organisation on the efficiency and effectiveness of AK retrieval. The analysis that we conducted and the correlation that we found show that it is possible to improve AK retrieval from SA documentation by providing more fitting AK organisation for the questions of documentation users.

Use of fitting AK organisation helped document users to quickly identify the location of answers and correctly recognize the answers. Lack of fitting AK organisation introduces search uncertainty about the location and completeness of answers, which caused document users to waste time searching for AK in wrong locations and miss answers. Ontology-based documentation can improve AK retrieval by providing a more fitting AK organisation for many questions, with more diverse possibilities to use the fitting AK organisation via multiple navigation paths to the AK that needs to be retrieved.

The file-based document organisation reflects a single-dimensional AK organisation that curtails efficient and effective AK retrieval. However, not all questions about AK are answered more efficiently and effectively using the ontology-based approach. We tested a subset of questions in our experiment, and these questions are about relationships between AK. A linear file-based AK organisation can be fitting for certain questions and it might not be cost-effective to provide ontology support for questions that are not often asked and for projects with few AK users.

We obtained similar results from experimentation in two companies which suggests promising results for using semantic wikis in an industrial setting. The cost-benefit analysis also indicates a positive return on investment.

7

Supporting Architecture Documentation: A Comparison of Ontologies for Knowledge Retrieval

In this chapter, we investigate how different ontology-based AK organisations influence the efficiency and effectiveness of AK retrieval from ontology-based documentation (RQ5). It is often difficult for SA document users to find all the AK they need to do their tasks, and this results in wasted time and mistakes during development. In this chapter we investigate how ontology-based documentation may support users in finding the AK they need. We executed a controlled experiment to test for differences in AK retrieval efficiency and effectiveness between ontologies built from different understandings of the AK needs of document users. We found that an improved understanding of AK needs allows for the construction of an ontology from which document users retrieve knowledge more efficiently and effectively. In constructing the ontologies, we applied ontology design criteria suggested by Gruber to improve their general qualities. In some cases we found that the ontology support for AK needs had to be traded off against ontology design criteria.

7.1 Introduction

Ontology-based documentation makes use of ontologies for non-linear organisation of AK via classes and relationships, and this allows document writers to comprehensively organise AK and relationships between AK for the needs of document users. Recent studies provide evidence that the use of ontology-based AK

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

organisation improves architectural review quality [51] as well as the efficiency and effectiveness of AK recovery [68], compared to file-based AK organisation. The previous Chapter 6 gives evidence that the use of ontology-based AK organisation improves the efficiency and effectiveness of AK retrieval, compared to the use of a file-based AK organisation.

However, we do not know how different ontologies perform in terms of their relative efficiency and effectiveness. In Chapter 6 we used ontologies built based on the expected AK needs, namely the lightweight software ontology from [103] which was built based on expected use cases for finding requirements and design, and the ontology built based on the typical AK needs of Océ professionals (see Chapter 5 and Figure 4.1). If we know the actual AK needs, and build an ontology according to these needs, how would this ontology perform? Would an ontology built from actual AK needs result in more efficient and effective AK retrieval than an ontology built from expected AK needs?

As such, we compare an ontology built from expected AK needs with an ontology built from actual AK needs. We investigate if there is any difference between the two ontologies in terms of AK retrieval efficiency and effectiveness. We also investigated the limitations of building an ontology from actual AK needs.

We report on a controlled experiment in which participants answer questions about AK from ontology-based SA documentation in a semantic wiki. The participants were organised in two groups. One group retrieved AK using an ontology built from the AK organisation in an SA document. Another group retrieved AK using an ontology built from the AK needs in an architectural review approach. Both ontologies were intended and used for architectural review[11], however, one was built from the *expected* AK needs of document users, i.e., built without knowing their exact AK needs and questions, and the other was built from the *actual* AK needs of document users in the experiment. We compared how the use of the two different ontologies affected the time-efficiency and effectiveness (in precision and recall) of AK retrieval by participants.

The experimental results show that the use of the ontology built from the actual AK needs of document users was significantly more efficient and effective for some experiment questions compared to the use of the ontology built from expected AK needs of users. We identified which parts of the ontology-based AK organisation supported the experiment questions. By analysing the search actions of experiment participants we verified that the use of supporting AK organisation positively correlates with the efficiency and effectiveness of AK retrieval.

We compared the AK organisation of the two ontologies, and found that the efficiency and effectiveness of AK retrieval was improved when supporting knowledge organisation was added. Adding redundant AK organisation did not significantly

improve AK retrieval efficiency and effectiveness. We used several criteria for designing clear and extendible knowledge sharing ontologies, and the use of these design criteria limited the support for AK needs in the constructed ontologies. The use of the ontology design criteria was thus traded off with lower AK retrieval efficiency and effectiveness.

This chapter makes the following contributions:

- Demonstrate how an improved understanding of AK needs can be used to provide ontology support for more efficient and effective AK retrieval.
- Identify the kind of ontology-based AK organisation that improves AK retrieval efficiency and effectiveness.
- Illustrate how the use of ontology design criteria affects the organisation and retrieval of AK.

Section 7.2 details on the experiment and its results. In Section 7.3 we analyse the ontology-based AK organisation to explain the underlying causes for the experiment results. In Section 7.4 we discuss insights gained from constructing the ontologies and analysing the experiment results. Section 7.5 describes threats to validity and Section 7.6 reports our conclusions and future work.

7.2 AK Retrieval Experiment

SA documentation often does not support the AK needs of all document users in its AK organisation [80]. It is hard for document writers to accurately predict all AK needs of document users, and this introduces a mismatch between the AK needs that are supported in an SA document and the actual AK needs of document users. We investigate whether support for the actual AK needs of document users in an ontology affects the efficiency and effectiveness of AK retrieval from ontology-based SA documentation.

We execute an experiment in which participants use two different ontologies to answer questions about AK as part of an architecture review. One ontology is built based on the AK needs that document writers expect document users to have (an 'expected-needs ontology') when conducting the architectural review. The other ontology is built based on the actual AK needs of document users (an 'actual-needs ontology') when they conduct the architectural review. The actual AK needs were identified by analysing questions in the architectural review approach, and modelled in the actual-needs ontology to provide an organisation for retrieving this AK.

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

We test whether there is a significant difference in time-efficiency and effectiveness (in recall and precision) of AK retrieval between the two ontologies.

The primary experimental goals are:

- **(A)** evaluate the AK retrieval **efficiency** of an ontology built based on expected AK needs and an ontology built based on actual AK needs.
- **(B)** evaluate the AK retrieval **effectiveness** of an ontology built based on expected AK needs and an ontology built based on actual AK needs.

The experiment took place during a software architecture course given at the University of Amsterdam. This course is part of a professional software engineering master programme. The experiment was advocated to students during one of the final lectures as a voluntary extracurricular activity, and participation did not affect their course grades. In total 49 students participated in the experiment.

7.2.1 Experiment Materials

We used a file-based SA document for constructing the ontology-based documentation. This file-based SA document is the main deliverable of one team of 5 students in the software architecture course.

The SA document describes the architecture of a social network system for software developers that can answer queries of individual software developers and analyse the development community. Several students acted as stakeholders with specific concerns and requirements for the system. Students in the architect role had to design the system such that it addresses the concerns and requirements.

The SA document is written in English, consists of 39 pages, 16 (mostly UML) diagrams, 528 paragraphs, and 10.874 words. The AK in this document is organised by a table of contents with 26 (sub)sections. The document has a view-based AK organisation, containing a logical & implementation view and deployment view, following the architecture description principles of Bass *et al.* in [9] (advocated as textbook for the course), as well as [57, 2].

The content of the SA document was imported as wikipages in ArchiMind by following the process described in Section 4.3. AK elements and relationships between AK elements described in wikipage content were subsequently annotated using the two ontologies (expected-needs and actual-needs ontology) introduced in the next subsections.

Ontology Design Criteria

We followed design criteria for knowledge sharing ontologies, proposed by Gruber in [42], when constructing the two ontology used in the experiment. These criteria aim for construction of clear and extendible ontologies that are suitable for sharing knowledge in different applications.

Design criterion 1) on '*Clarity*' states that ontology constructs should be clear and objective, e.g., supported by natural language descriptions to specify their meaning.

Design criterion 2) on '*Coherence*' states that an ontology should be logically consistent, i.e., that its constructs and axioms do not contradict each other.

Design criterion 3) on '*Extendibility*' states that ontology extensions should be anticipated to support future tasks, and not require revision of existing ontology constructs.

Design criterion 4) on '*minimal encoding bias*' states that the definition of concepts should not depend on a particular encoding, implementation, or notation, because this restricts ontology use to specific programs and tools. Our use of ontology class '*Wikipedia*', for storing SA document content in ArchiMind, is a form of encoding bias.

Design criterion 5) on '*minimal ontological commitment*' states that an ontology should model a domain using the minimal set of constructs and claims necessary for its knowledge sharing activities. This allows more freedom when extending and instantiating the ontology in different and specialized domains.

Expected-needs Ontology

We used the file-based SA document to construct an ontology for organising and indexing its AK in ontology-based documentation. We refer to this ontology as an '*expected-needs ontology*' because it was constructed based on the AK needs that the document writers *expected* document users to have. The expected-needs ontology is depicted in Figure 7.1 (without grey elements).

The 5 students who wrote the SA document knew that their architecture would be reviewed by the course staff for grading, and reviewed by other students as part of an assignment. The document writers expected architectural review by document users, and they understood that the AK needed for these review tasks should be recorded in their SA document. Similarly, the two researchers that built the ontology knew that the SA document was written with architectural

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

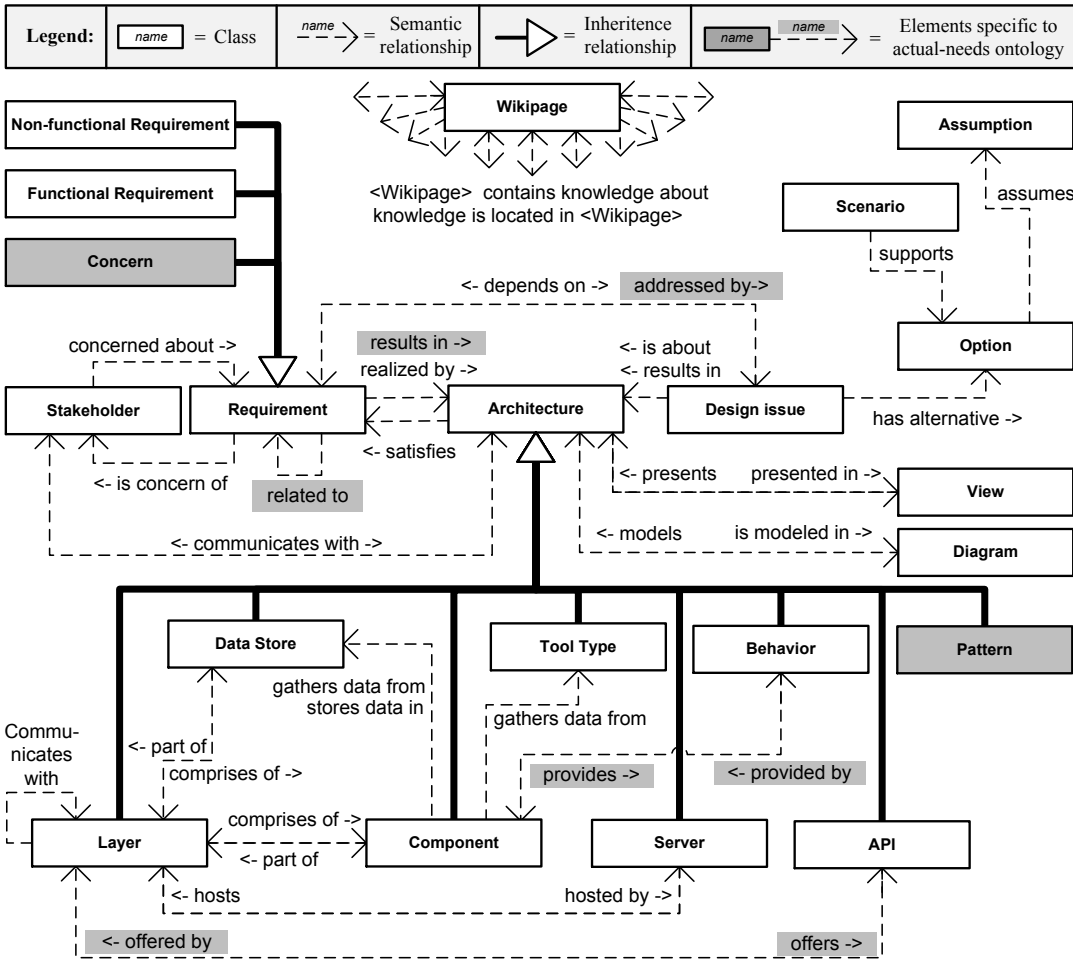


Figure 7.1: Expected-needs ontology (without grey elements) and actual-needs ontology (all elements)

review (by course staff and students) in mind. The researchers and the document authors however did not know the exact AK needs and actual questions that had to be answered during review.

The expected-needs ontology employed in the experiment was constructed based on the AK organisation of the file-based SA document, using the table of contents, views, tables, diagrams, lists of AK elements, and boldface headers, which make explicit where different types of AK and relationships between AK are recorded.

Two researchers collaborated to identify AK concepts and relationships between AK based on the available AK organisation and their understanding of SA concepts.

For example, the table of contents in the SA document has subsections "*design rationale*", "*assumptions*", and "*scenarios*", which makes it explicit where design rationale is recorded. Boldface text and table column headers made design options explicit in the content of these sections. Based on this AK organisation we modelled class '*Design issue*', '*Option*', '*Assumption*', and '*Scenario*', as well as relationships '*assumes*' and '*supports*' in the expected-needs ontology, as depicted in Figure 7.1. Various UML diagrams made elements of the architectural solution design explicit, e.g., components and layers, which were modelled as subclasses of '*Architecture*' in the ontology. In [51] a similar process was used to derive a domain ontology, during which AK elements were annotated in SA document content.

We followed the five design criteria, introduced in Section 7.2.1, whilst constructing the expected-needs ontology. Most notably, we modelled class '*Option*', '*Assumption*', and '*Scenario*', instead of a single class '*Decision*', when considering design criterion 3. This allows for an *extended* definition of design rationale and supports future associated tasks, e.g., extensive evaluation of design rationale, without having to revise the ontology.

Actual-needs Ontology

In [76] Nord *et al.* provide a number of questions, organised in question sets, to review architecture documentation as part of an SEI architectural review approach. 50 of the 127 SEI questions could be fully or partially answered from the SA document used in the experiment.

We used the AK needs expressed in these 50 SEI questions to build the actual-needs ontology. We identified AK types and relationships between AK in the SEI questions, and added these AK types and relationships to the actual-needs ontology if they were not yet present in the expected-needs ontology. This aims to support the AK needs of document users (experiment participants) when they apply the SEI architectural review approach.

For example, we added relationships '*provides*', '*offers*', and class '*Pattern*' to support document users in answering SEI questions about software development activities. We also added an ontology class '*Concern*' to support SEI questions for reviewing the documentation of stakeholders, concerns, and conformance to ISO/IEC 42010 [2].

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

The actual-needs ontology extends the expected-needs ontology. We chose for ontology extension, instead of building another ontology from scratch, because this allows us to investigate how the addition of specific ontology constructs to support AK needs influences AK retrieval. The added ontology constructs are marked grey in Figure 7.1.

The above process is in part similar to that in the 'typical question' approach to ontology engineering for software architecture documentation, which is described in Chapter 5 and [26]. In [26], questions that SA document users ask during their daily tasks, i.e., their 'typical questions' that represent their AK needs, are used to construct an ontology that supports the AK needs of document users.

We again used the ontology design criteria by Gruber [42] (see Section 7.2.1). For example, several of the SEI questions that review the support for software development activities required retrieval of implementation units and development dependencies. We however decided not to model implementation units and development dependencies as AK types and relationships in the ontology because this introduces a lot of *ontological commitment* (design criterion 5).

Several SEI questions for reviewing the documentation of stakeholders aim to locate stakeholder concerns in the SA document content, which is stored in wikipages in ArchiMind. We however decided to not add more relationships to and from class 'Wikipedia' because Wikipedia is application specific and introduces *encoding bias* (design criterion 4).

We added relationship 'addressed by', between class 'Requirement' and 'Design issue', as well as relationship 'results in', between class 'Requirement' and 'Architecture' in the actual-needs ontology. This supports SEI questions for reviewing the identification of requirements and design decisions. The added relationships are more specific than the existing relationships in the expected-needs ontology, and this adheres to design criterion 1) *clarity*.

Experiment Questions

We selected 5 SEI questions which are representative of the AK needs in the 50 SEI questions that were used to build the actual-needs ontology. These 5 SEI questions are used as experiment questions to test and compare AK retrieval efficiency and effectiveness between the two ontologies. We used several criteria to select the 5 experiment questions from the 50 SEI questions.

Firstly, the experiment questions must be answerable from the SA documentation used in the experiment. Secondly, the experiment questions must have answers that can be quantitatively assessed, i.e., such that evaluators do not have to

subjectively judge whether the answer to an open-ended question is either correct or incorrect. Thirdly, selected experiment questions should be representative of multiple SEI questions, to cover a large part of the SEI architectural review approach. We finally selected the following questions:

- 1: *List ten development dependencies between implementation units.*
- 2: *Which implementation units and decisions are explicitly related to requirement 'Security'?*
- 3: *Which architectural patterns are described in the architecture?*
- 4: *Which functional requirements are related to non-functional requirement 'Compatibility'?*
- 5: *Find ten wikipages in which the concerns of the stakeholders are addressed (not just listed).*

The experiment questions cover several aspects of the SEI architectural review approach. Question 1 reviews the support for software development in an SA document. Question 2 reviews the support for comprehensive architectural evaluation, software development, and identification of requirements and decisions in an SA document. Question 3 reviews the support for software development, and question 4 reviews the support for identification of requirements and design decisions. Question 5 reviews conformance to ISO/IEC 42010, support for comprehensive architectural evaluation, and reviews whether important stakeholders and concerns are captured.

We estimated that it was not acceptable for all student participants to spend more than 45 minutes on the experiment. Therefore we limited the number of answers for experiment questions 1 and 5 to ten, and instantiated the requirements that have to be retrieved in questions 2 and 4. The original SEI questions require complete AK retrieval and thus more time. Question 3 is a shorter version of an SEI question.

7.2.2 Experiment Hypothesis

We formulate the following alternative hypotheses for experimental goals A and B specified in Section 7.2;

H_{1A} = *The use of the actual-needs ontology for answering experiment questions results in higher time-efficiency than the use of the expected-needs ontology.*

H_{1B} = *The use of the actual-needs ontology for answering experiment questions results in higher effectiveness than the use of the expected-needs ontology.*

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

The null hypotheses state that there is no difference in efficiency and effectiveness between the use of the two ontologies.

In the experiment we used one independent variable (or ‘predictor variable’) with two levels, namely the expected-needs and the actual-needs ontology. Two dependent variables (or ‘response variables’) are used in the experiment. **Time** is used as a measure of efficiency. The harmonic mean of precision and recall, the **F1 score**, introduced by van Rijsbergen in [114], is used for measuring effectiveness. See Section 6.1.4 in previous chapter for an explanation of the F1 score.

The students that wrote the SA document also participated in the experiment. Their answers to the experiment questions were only used to verify the relevancy of items, or ‘ground truth’, when calculating precision and recall.

7.2.3 Experiment Procedure

A ten-minute introduction to ontology-based SA documentation was given to participants during the final lecture of the software architecture course. A researcher explained the functions and GUI of ArchiMind using a graphical overview in presentation slides. The participants also received a printed form with a graphical overview of ArchiMind’s GUI and an explanation of its functionality.

The participants received another printed form which depicted the ontology they had to use. This form also instructed participants: "*Please try to simulate your behavior in normal work and studies: balance between time-efficiency and correct answers.*". This instruction was given to encourage participants to perform the experiment seriously and answer the questions as time-effectively as possible.

We handed out two versions of the printed experiment instructions in random order. In one version of the instructions we asked participants to navigate to a URL which hosted ArchiMind with the expected-needs ontology. The other version led participants to a URL which hosted ArchiMind with the actual-needs ontology. These URLs also hosted a web-based form which listed the questions, input boxes for answers, and the expected format of answers. The experiment took place in two classrooms with PCs containing the same hardware and operating system image.

Before the start of the experiment we asked each participant: "*How many years of working experience in IT industry do you have?*". Participants reported an average of 3.92 years experience in IT industry. Participants using the expected-needs ontology have 4.78 years experience on average and participants using the actual-needs ontology reported 2.87 years of experience on average. Two participants using the expected-needs ontology were outliers with 16 and 30 years

7.2. AK RETRIEVAL EXPERIMENT

of experience, which explains the large difference in average years of experience between the two groups.

Table 7.1: Time-Efficiency (Seconds), Effectiveness (F1 Score), and Statistical Test Results in Experiment

Ex- peri- ment ques- tion	Metric (for each table row)	Av- erage expect- ed- needs	Av- erage actual- needs	Differ- ence	<i>p</i> - value test re- sults	Num- ber of mea- sure- ments	Effect size <i>r</i>
1	Seconds	1094	1278	184	0.26102	40	0.18
	F1score	0.22	0.32	0.10	0.34867	43	0.14
2	Seconds	349	376	27	0.53151	36	0.10
	F1score	0.72	0.63	0.09	0.11478	38	0.26
3	Seconds	246	117	129	0.00082	30	0.61
	F1score	0.24	0.79	0.56	0.00010	34	0.67
4	Seconds	290	229	61	0.39308	27	0.16
	F1score	0.08	0.75	0.67	0.00003	33	0.73
5	Seconds	448	366	82	0.41236	24	0.17
	F1score	0.72	0.60	0.12	0.23968	29	0.22

7.2.4 Experiment Test Results

Using the Shapiro-Wilk and Kolmogorov-Smirnov tests, we found that the experiment measurements were not normally distributed. Therefore we applied the non-parametric Mann-Whitney-Wilcoxon (MWW) test to the experiment measurements. Table 7.1 reports the average efficiency and effectiveness measurements, the *p-values* for one-tailed MWW tests (corrected for ties), and effect size per question.

12 participants encountered errors in ArchiMind whilst answering questions (the database server could not handle 49 concurrent users). We excluded measurements for these questions. In some cases we could include the F1 scores of answers in our measurements because participants lost time due to an error, but could still continue to give an answer. This results in an unpaired number of measurements (see second-to-last column of Table 7.1) between the control and test group, however, the MWW test allows testing of unpaired measurements.

Column ‘*p-value test results*’ in Table 7.1 contains the *p-values* for one-tailed

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

MWW test results on efficiency for each row with ‘*Seconds*’ in column ‘*Metric*’. Table 7.1 shows that the difference in AK retrieval efficiency between the expected-needs and actual-needs ontology is statistically insignificant at the $p=0.05$ level for all experiment questions, except for question 3. Consequently, we only reject the null hypothesis \mathbf{H}_{0A} and accept the alternative hypothesis \mathbf{H}_{1A} for question 3. Most participants that used the expected-needs ontology quickly gave up searching without finding answers to question 4. This explains the insignificant difference in efficiency between the ontologies for question 4.

Table 7.1 shows that the difference in AK retrieval effectiveness (rows with “*F1score*”) between the expected-needs and actual-needs ontology is statistically significant at the $p=0.05$ level for experiment questions 3 and 4. Consequently, we reject the null hypothesis \mathbf{H}_{0B} and accept the alternative hypothesis \mathbf{H}_{1B} for questions 3 and 4.

7.3 AK Organisation and AK Retrieval

In this section we analyse the experiment results. The actual-needs ontology provided significantly higher AK retrieval efficiency and effectiveness than the expected-needs ontology for experiment questions 3 and 4. The only intended (or ‘designed’) difference in the experiment materials was between the AK organisations provided by the two ontologies. We explain the differences in AK retrieval efficiency and effectiveness by comparing the difference in AK organisation that the two ontologies provided.

In Section 7.3.1 we analyse which ontology-based AK organisation supported participants in finding the AK and relationships between AK for each experiment question. We analyse the usage of supporting AK organisation from the search actions of participants in Section 7.3.2 and verify that this usage leads to efficient and effective AK retrieval. Section 7.3.3 describes how the AK retrieval efficiency and effectiveness of participants was affected by the supporting AK organisation for each experiment question and by different ontology constructs.

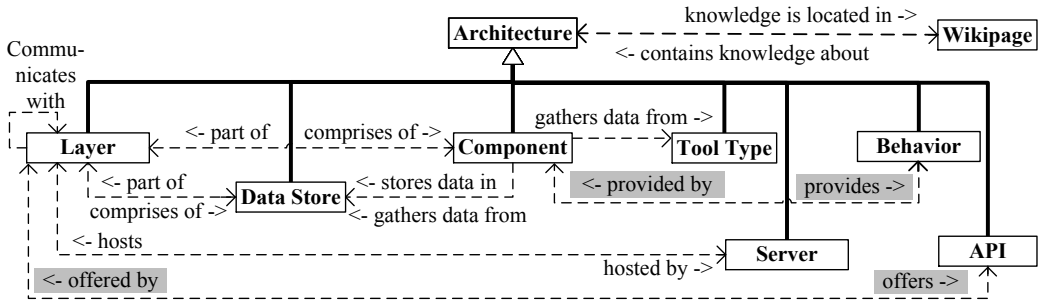
7.3.1 Fitting AK Organisation

The ontology-based documentation tested in the experiment is organised by ontology classes and relationships between classes. Figure 7.2 depicts the ontology-based AK organisation that provided one or more paths to the answers for each experiment question, i.e., the ontology classes with AK instances and the re-

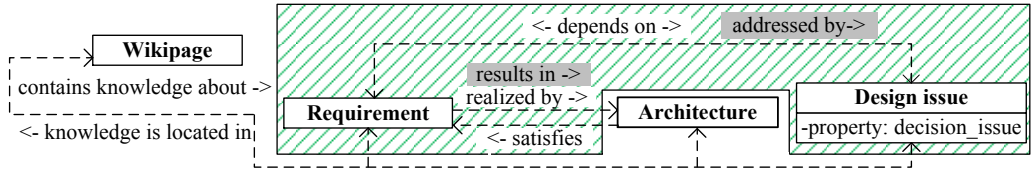
7.3. AK ORGANISATION AND AK RETRIEVAL

Legend	non-fitting AK organisation	fitting AK organisation
Ontology elements in both expected-needs and actual-needs ontology	class — Relationship ->	class — Relationship ->
Ontology elements only in actual-needs ontology	class — Relationship ->	class — Relationship ->

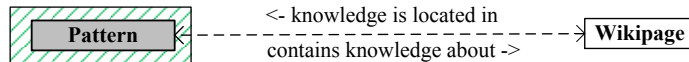
Question 1: List ten (10) development dependencies between implementation units.



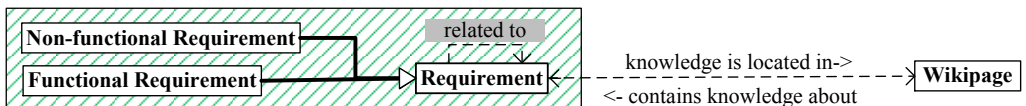
Question 2: Which implementation units and decisions are explicitly related to requirement 'Security'?



Question 3: Which architectural patterns are described in the architecture?



Question 4: Which functional requirements are related to non-functional requirement 'Compatibility'?



Question 5: Find ten (10) Wikipages in which the concerns of the stakeholders are addressed (not just listed).



Figure 7.2: Ontology-based AK organisation that provided paths to answers for experiment questions

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

relationships between classes that can be used to navigate to AK instances and answers.

Some of the nodes on a path to the answer explicitly relate to the AK needs in the question asked. For example, experiment question 4 “*Which functional requirements are related to non-functional requirement 'Compatibility'?*”, specifies that instances of AK types ‘*functional requirement*’ and ‘*non-functional requirement*’ have to be found. Both the actual-needs and expected-needs AK organisation in Figure 7.2 contain classes ‘*Functional Requirement*’ and ‘*Non-functional Requirement*’ which relate to the two AK types in question 4, and these classes can be used to find a path to answers for question 4.

Experiment question 4 also specifies a relationship ‘*related to*’ between requirements, which is not supported in the expected-needs AK organisation. Participants can still identify the relationships between requirements in the content of wikipages, from properties of class *Requirement*, or by deriving the relationships via transitive or symmetric properties of other ontology relationships to and from requirements, e.g., ‘*depends on*’, ‘*is concern of*’, and ‘*realized by*’. The AK organisation provided by the actual-needs ontology additionally contains relationship ‘*related to*’, which explicitly matches ‘*related to*’ in experiment question 4 and provides a direct path to its answers.

We term the elements in the ontology-based AK organisation that explicitly refer to the AK needs in the question asked, and provide a direct path to its answer(s), as “*fitting AK organisation*”. We adopt the specific term “fitting AK organisation” and its definition because there may be other forms of support that an ontology-based AK organisation provides for AK needs in questions. Ontology classes and relationships that provided fitting AK organisation for AK needs in an experiment question are surrounded by boxes filled with green diagonal lines in Figure 7.2.

7.3.2 Usage of Fitting AK Organisation

There are various (confounding) factors that may have influenced the efficiency and effectiveness of AK retrieval in the experiment. For example, participants could find answers by keyword searching and reading plain text stored in wikipages, without making use of the fitting AK organisation provided by the ontologies. In this section we verify that the usage of fitting AK organisation influences the efficiency and effectiveness of AK retrieval.

The verification approach in this section is largely the same as the approach reported in previous chapter in Section 6.2.2. Please see Section 6.2.2 for a more detailed explanation of the verification approach.

7.3. AK ORGANISATION AND AK RETRIEVAL

We quantified the usage of AK organisation from over 3,000 AK retrieval actions of participants that were automatically logged during the experiment. Each logged search action is time-stamped and contains the name of the search action (e.g., 'keyword search' and 'class navigation'), the AK organisation that was used (e.g. class 'Requirement'), and the names of visited AK instances or keyword search terms.

We measured the use of fitting AK organisation if 1) the fitting AK organisation appeared on the screen of a participant for 3 seconds or more, and 2) the participant navigated to answers by following the fitting AK organisation or gave answers based on the fitting AK organisation shown on screen. We evaluated these criteria by looking at the properties of the logged search actions, their time-stamp, and by re-enacting the logged search actions in ArchiMind.

We measured usage of the fitting AK organisation based on the ontology classes and relationships, that provided fitting AK organisation for AK needs, depicted in Figure 7.2. This includes the use of fitting AK organisation in wikipage content, which was annotated based on the ontologies (see Section 4.3 for details on the annotation mechanism).

RatioTimeFitting is introduced as a metric to represent how much fitting AK organisation was used to answer an experiment question. *RatioTimeFitting* is calculated per participant per experiment question by dividing the 'time spent using fitting AK organisation' by the 'total time spent searching for AK'. *Time-effectiveness* is introduced in order to represent the efficiency and effectiveness of AK retrieval in a single metric. *Time-effectiveness* is calculated per answer in the experiment by dividing the *F1 score* (effectiveness) by the 'total time spent searching for AK' (efficiency).

To verify that the use of fitting AK organisation influences the efficiency and effectiveness of AK retrieval, we test if a correlation exists between *RatioTimeFitting* and *Time-effectiveness*. We use the following hypothesis:

H_{1C} = *There is a correlation between RatioTimeFitting and Time-effectiveness.*

The null hypothesis states that there is no correlation.

Using the Kolmogorov-Smirnov test, we found that the measurements for *RatioTimeFitting* and *Time-effectiveness* are not normally distributed. Therefore the non-parametric Spearman's rank correlation test is applied.

Application of Spearman's rank test indicates a moderate positive correlation (coefficient $r = 0.509$) between *RatioTimeFitting* and *Time-effectiveness*. Figure 6.5 depicts a similar correlation in the Océ and LaiAn experiment.

These test results are statistically significant at the $p=0.01$ level with a (2-sided)

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

P-value of 1.005^{-11} . Consequently, we reject the null hypothesis H_{0C} and accept the alternative hypothesis H_{1C} . This shows that the use of fitting AK organisation was positively correlated with the time-effectiveness of AK retrieval.

7.3.3 Fitting AK Organisation per Question

The experiment results in Table 7.1 show that there was no significant difference in AK retrieval efficiency and effectiveness between the two ontologies for questions 1 and 2. The insignificant difference for question 1 can be explained by the lack of fitting AK organisation in both ontologies, as shown in Figure 7.2. The actual-needs ontology contained four additional relationships that provided paths to answers for questions 1. The names of the four relationships did however not make it explicit to users where they could find development dependencies, and were thus not fitting for question 1.

The actual-needs ontology contained two additional relationships that provided fitting AK organisation for the AK needs in question 2. However, these two additional relationships in the actual-needs ontology were redundant because existing relationships '*depends on*', '*realized by*', and '*satisfies*' in both ontologies provided the same paths to answers and were fitting for the same AK needs in question 2. The actual-needs ontology thus provided redundant fitting AK organisation for AK needs in question 2, and its use did not result in significantly higher AK retrieval efficiency and effectiveness compared to use of the expected-needs ontology.

The actual-needs ontology provided additional fitting organisation by means of ontology class '*Pattern*' for AK needs in question 3 and relationship '*Related to*' (between requirements) for AK needs in question 4, as depicted in Figure 7.2. These AK needs were not yet supported by fitting AK organisation in the expected-needs ontology. Consequently, the AK retrieval efficiency and effectiveness of the actual-needs ontology was significantly higher than that of the expected-needs ontology for question 3, and AK retrieval effectiveness was significantly improved for question 4. This evidence also shows that a single class or relationship can be added to an ontology to provide fitting organisation for AK needs and thereby improve the time-effectiveness of AK retrieval.

The actual-needs ontology contains an additional class '*Concern*', which provided fitting AK organisation for AK needs in question 5. However, the expected-needs ontology already provided fitting AK organisation to support participants in finding the concerns for question 5, namely, via relationships '*is concern of*' and '*concerned about*'. Consequently, the AK retrieval efficiency and effectiveness of the actual-needs ontology was not significantly higher than that of the expected-

needs ontology. Adding fitting AK organisation that is redundant to existing fitting AK organisation thus did not help to improve AK retrieval efficiency and effectiveness.

7.4 Discussion

We built the expected-needs ontology based on the AK organisation of the SA document that was used in the experiment. The SA document contained all the AK needed by participants in the experiment because we only asked questions that could be answered from the SA document content. Moreover, the document authors anticipated the AK needs for architectural review, and architectural review questions were asked in the experiment. As such the SA document contained all AK needed in the experiment, which could have been modelled in the expected-needs ontology to provide fitting AK organisation.

However, certain types of AK and relationships between AK were not very explicit in the SA document, and difficult to identify without knowing if this AK was needed by document users. The AK in the SA document was organised to support the anticipated AK needs of its users, yet it did not clearly specify these AK needs. Building the expected-needs ontology thus required reverse engineering of the AK needs for which the SA document was written.

We noticed that the relationships between AK were especially hard to identify because they were not always explicitly described in the SA document. For example, the table of contents and boldface headers made it very explicit where decisions were recorded, yet their relationships with the requirements were less explicitly listed in the text inside the sections. This issue might be addressed by assuming that certain relationships between AK exist, instead of identifying and modeling the relationships between AK that are very explicit in the SA document.

We constructed the actual-needs ontology using the coding and ontology construction step in the 'typical question' approach described in Chapter 5. In Chapter 5 we suggest that the questions of users help AK ontology construction because they clearly identify the required types of AK and relationships between AK. This suggestion is supported by the study in this chapter since we were able to identify AK types and relationships between AK from the SEI questions to construct the actual-needs ontology.

The actual-needs ontology however did not provide fitting AK organisation for all AK needs in the experiment, even though AK needs were identified from SEI questions. This lack of fitting AK organisation curtailed the efficiency and effectiveness of AK retrieval. The correlation found in Section 7.3.2 suggests that

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

participants would have retrieved AK more efficiently and effectively if additional fitting AK organisation was provided to support their AK needs.

We found that the use of the ontology design criteria (see Section 7.2.1) restricted the modelling of the identified AK needs. The design criteria helped to construct a clear and extendible ontology for knowledge sharing across different applications. For example, based on these criteria we decided to comprehensively model design rationale to support future AK retrieval tasks (see Section 7.2.1). We also decided not to model certain constructs in the actual-needs ontology because they affected ontological commitment and encoding bias (see Section 7.2.1). This allows for more freedom to use the ontology in different domains and applications.

However, these decisions, made based on the ontology design criteria, restricted modelling of fitting AK organisation to support AK needs, and this in turn negatively affected the AK retrieval efficiency and effectiveness of participants. Use of the ontology design criteria was thus traded off against lower efficiency and effectiveness of AK retrieval.

Above findings can be used in practice to create ontology-based SA documentation from which AK can be retrieved efficiently and effectively. In [36] Falessi *et al.* investigated whether an accurate understanding of the value of AK for the activities of document users can be used to only document valuable AK, and thereby reduce the costs of producing documentation. Conversely, we investigated whether an accurate understanding of the AK needs of document users can be used to improve AK retrieval from documentation, and thereby reduce time-costs and increase the effectiveness of AK retrieval when consuming documentation.

7.5 Threats to Validity

Internal Validity

The file-based SA documentation which was used as input for the experiment was written by 5 students, of which 4 participated in the experiment. We excluded the answers of these document authors from statistical testing because their prior knowledge about the SA documentation would bias the experiment results. Instead we used their answers as a 'gold standard' (i.e. 'ground truth') to verify that our evaluation of precision and recall was correct.

A group of 5 students who acted as stakeholders in the SA course answered several SEI questions from the file-based SA document as part of an assignment prior to the experiment. The AK needs in these questions partially overlaps with the AK

needs in experiment questions, which means that these 5 students may have had prior knowledge about the answers to questions in the experiment. This threat to validity was mitigated by equal distribution of the 4 stakeholder-students that participated in the experiment across the test and control group: two students used the expected-needs ontology and two used the actual-needs ontology.

Two students registered as 'anonymous' for the experiment, and may have been part of the 5 stakeholders or 5 document authors discussed above. They however retrieved AK with average efficiency and effectiveness, which leads us to believe they do not have prior knowledge and are not part of the document authors or stakeholders discussed above.

External Validity

The specific set of questions from the SEI architectural review approach that were used in the experiment limits generalization of the experiment results and findings. The SA documentation used in the experiment was written by students that used guidelines in [57, 9, 2] for view-based architecture description. The experiment documentation can be generalized to documentation practice in industry to the extent that industry practitioners also make use of UML and view-based architecture descriptions.

The ArchiMind semantic wiki tool is not specific to the ontologies and SA documentation in the experiment, and can be used for any type of ontology and documentation. The findings in this work can to a certain extent be generalized to ontology-based documentation approaches that use a semantic wiki.

7.6 Conclusions

It is important that AK is quickly and correctly retrieved from SA documentation, however, SA documentation often does not support its users in retrieving all the AK that they need for their tasks. In this chapter we investigated whether document users retrieve AK more efficiently and effectively when they use an ontology-based document organisation that is constructed from an improved understanding of their AK needs. We conducted an experiment to compare AK retrieval using an ontology built from the expected AK needs of document users and an ontology built from their actual AK needs.

The use of the actual-needs ontology was significantly more efficient and effective for answering one experiment question, and significantly more effective for

CHAPTER 7. SUPPORTING ARCHITECTURE DOCUMENTATION: A COMPARISON OF ONTOLOGIES FOR KNOWLEDGE RETRIEVAL

answering another experiment question, compared to the use of the expected-needs ontology. Part of the ontology-based AK organisation was fitting for AK retrieval; it explicitly denoted the AK types and relationships between AK that participants needed to retrieve. We found that the use of AK organisation that was fitting for AK needs in a question had a significant correlation with the efficiency and effectiveness of answering that question. The actual-needs ontology provided more fitting AK organisation for the AK needs in two questions, and its users answered one of the two questions more efficiently and effectively and the other question more effectively than users of the expected-needs ontology.

The experiment results show that a better support for the AK needs of ontology-based documentation users improves the efficiency and effectiveness when the users retrieve the AK that they need. The results highlight the importance of accurately understanding the AK needs of SA documentation users; it allows document writers to organise AK such that document users can efficiently and effectively retrieve the AK that they need.

Not all AK needs were supported by fitting AK organisation, and this curtailed the efficiency and effectiveness of AK retrieval. This was caused by the use of criteria for designing clear and extendible knowledge sharing ontologies, which limited the amount of fitting AK organisation that was added to support AK needs. The use of the ontology design criteria was traded off against the efficiency and effectiveness of AK retrieval. In future studies we plan to investigate how fitting AK organisation can be provided for more AK needs of document users, whilst using design criteria for knowledge sharing ontologies.

8

Conclusions

In this thesis we investigated whether the efficiency and effectiveness of AK retrieval can be improved using ontology-based documentation. We first studied how AK is retrieved from file-based documentation in practice and theory. We then introduced an ontology-based approach for retrieving AK from SA documentation. Next, we proposed and applied an approach to build an ontology for SA documentation in a software project. We conducted experiments to compare the efficiency and effectiveness of AK retrieval between file-based and ontology-based documentation. Finally we compared AK retrieval between two ontologies in ontology-based documentation. In this chapter we revisit the research questions, reflect on contributions, and discuss future work.

8.1 Contributions

Even a perfect SA is essentially useless if its AK cannot be found and understood [9, 21]. SA documentation should be written to satisfy the needs of its users, and organised such that users can quickly find the AK they need and answer their questions [21, 77]. It is common practice to capture AK in file-based documents [80], however, various challenges inhibit efficient and effective AK retrieval from these documents. Recent studies provide evidence that the use of ontology-based documentation improves AK extraction [51] and AK understanding [68] compared to file-based documentation.

Our main Research Question (RQ) is whether we can improve the efficiency and effectiveness of AK retrieval using an ontology-based documentation approach. In previous chapters we investigated RQ1 to RQ5. We summarize their answers in this chapter and discuss how together they provide an answer to the main RQ.

CHAPTER 8. CONCLUSIONS

In Chapter 2 and 3 we investigated RQ1; "*how do software professionals retrieve AK from file-based documentation?*". In Chapter 2 we used protocol analysis to investigate the search behaviour of professionals in industry and we identified four search strategies. We found that professionals experience search uncertainty and AK retrieval challenges when they cannot follow document organisation that relates to the questions they need to answer. Use of prior knowledge helps to deal with search uncertainty, however, it is prone to cognitive bias and often results in inefficient and ineffective AK retrieval.

We studied literature about file-based SA documentation in Chapter 3 and identified underlying causes for missing document organisation and AK retrieval challenges. Many of the challenges that are reported by software industry professionals stem from the linear organisation of AK in file-based documentation. This suggests that there is room for improving the efficiency and effectiveness of AK retrieval practices.

We proposed an ontology-based documentation approach when investigating RQ2; "*How can an ontology be used for retrieving AK from documentation?*", in chapter 4. We first discussed how ontologies can be used for organising and retrieving AK in SA documentation. We then described a lightweight software ontology and semantic wiki that are used for AK retrieval in the proposed ontology-based approach. The identified AK retrieval challenges in file-based documentation can be addressed by the ontology-based approach.

We proposed an ontology engineering approach when investigating RQ3; "*How to construct an ontology for SA documentation in a software project context?*", in Chapter 5. We conducted an exploratory case study to evaluate how well our approach works to construct an ontology, given the diversity of AK users, domain complexity, and other contextual factors in a software project. Industry practitioners evaluated the constructed ontology as useful for retrieving AK.

In Chapter 6 we investigated RQ4; "*How do file-based and ontology-based documentation influence the efficiency and effectiveness of AK retrieval?*". We conducted an experiment in two companies during which software professionals answered questions about AK. Use of ontology-based documentation was significantly more efficient and effective than use of file-based documentation.

Part of the available AK organisation was fitting for AK retrieval; it explicitly denoted the AK types and relationships between AK specified in the experiment questions. We analysed search actions of professionals and found that the usage of fitting AK organisation has a positive correlation with the efficiency and effectiveness of AK retrieval. This correlation explains the difference in efficiency and effectiveness between the two documentation approaches.

We conducted a questionnaire among industry professionals, which shows a mostly positive evaluation of ontology-based documentation. A coarse cost-benefit estimation indicates a positive return on investment when replacing file-based with ontology-based documentation in the studied projects.

In Chapter 7 we investigated RQ5; "*How do different ontology-based AK organisations influence the efficiency and effectiveness of AK retrieval?*". We compared AK retrieval from an ontology built based on the expected AK needs of document users, and from an ontology built based on the actual AK needs. The results show that a better understanding of AK needs allows for the construction of ontologies that provide more fitting AK organisation, and thereby improve AK retrieval efficiency and effectiveness.

Main Research Question

The findings in this thesis support a confirmatory answer to the main RQ; "*Can we improve AK retrieval efficiency and effectiveness using ontology-based documentation?*".

We phrase the findings using the same terminology as the main RQ, to summarize how the main RQ is answered.

- AK retrieval efficiency and effectiveness can be improved in existing file-based documentation practice in industry (RQ1).
- Ontology-based documentation can address challenges that inhibit efficient and effective AK retrieval in existing file-based documentation practice (RQ2).
- An ontology that is useful for retrieving AK in ontology-based documentation can be built in a software project (RQ3).
- The use of ontology-based documentation can improve AK retrieval efficiency and effectiveness compared to the use of file-based documentation in industry practice (RQ4).
- AK retrieval efficiency and effectiveness in ontology-based documentation can be improved by using ontology-based AK organisation that is built based on a better understanding of AK needs (RQ5).

Though the results of this study indicate that ontology-based SA documentation holds a promising future, there are still many challenges to overcome. More research is needed to replicate and further generalize the findings to different types of software projects, and to explore open questions and future work.

8.2 Innovative Aspects

The research in this thesis has several innovative aspects:

- Use of Protocol Analysis to identify and evaluate search strategies and heuristics during AK search in file-based documentation.
- Introduction of an ontology-based SA documentation approach that uses a semantic wiki.
- Introduction of an approach that uses Grounded Theory to construct ontologies for SA documentation in a software project context.
- Experimental evaluation and comparison of AK retrieval from ontology-based and file-based documentation.
- Quantification of AK organisation usage during AK retrieval from ontology-based and file-based documentation.
- A cost-benefit estimation of adopting ontology-based documentation in two software projects.
- Experimental evaluation and comparison of two ontologies for AK retrieval.

8.3 Discussion and Future Work

In the previous section we discussed how the findings in this thesis answer the RQs about ontology-based documentation. The findings not only provide answers to the RQs, but also provide additional or overreaching insights as well as directions for future research and industry practice. We discuss these insights and research directions per theme in the following subsections.

8.3.1 Organising AK for Efficient and Effective Retrieval

We found that professionals experience search uncertainty when the organisation of AK in documents does not relate to their questions about AK. They become uncertain about the completeness of answers when it is not clear from document and section titles where relevant AK is located. Removing uncertainty by reading all document content is time-consuming, and spelling variations, synonyms, and acronyms make keyword searching error-prone. Professionals that searched under uncertainty often retrieved AK inefficiently and ineffectively.

We also identified AK organisation that was fitting for AK retrieval; it explicitly denoted the AK types and relationships between AK specified in the questions that professionals had to answer. Use of fitting AK organisation positively correlated with the efficiency and effectiveness of AK retrieval. This suggests that AK retrieval from existing SA documentation in industry can be improved by providing more fitting AK organisation for the questions of document users.

In this thesis we identified AK organisation that is fitting for answering well-defined questions in file-based and ontology-based documentation. There might be other forms of fitting AK organisation, e.g, in other types of SA documentation, and for AK retrieval tasks without well-defined questions, e.g., learning about SA. Given the potential benefits for AK retrieval, we encourage empirical identification of fitting AK organisation in other documentation types, e.g., hypertext-based documentation in wikis, and for various AK retrieval tasks.

8.3.2 Understanding the AK Needs of Document Users

Understanding the AK needs of document users allows document authors to include and organise AK such that users can quickly find the AK they need, and this increases the value of SA documentation [21]. We found that an improved understanding of AK needs can be used to build a more fitting AK organisation and thereby improve AK retrieval efficiency and effectiveness. In this thesis we used several AK organisations, each based on a different understanding of the AK needs:

- **Use Cases:** The lightweight software ontology was built to support use cases that are representative of AK needs during typical activities of architects [103]. This ontology provided partially fitting AK organisation for LaiAn experiment questions.
- **Representative questions:** We built an ontology based on typical questions that represent AK needs in the Océ domain. This ontology provided largely fitting AK organisation for other representative questions in the Océ experiment.
- **Existing Documentation:** The file-based documents used in the Océ and LaiAn experiment were organised based on document authors' understanding of the AK needs. These documents contained partially fitting AK organisations for experiment questions. In Chapter 7 we built an ontology based on AK needs that we elicited from the AK organisation of a file-based SA document. This AK organisation is representative of the AK needs that the document authors knew or expected the document users to have. The ontology provided partially fitting AK organisation for architectural review questions. The linear organisation of

file-based documents in a table of contents limited the amount of fitting AK organisation that could be included, regardless of whether the document authors had an accurate understanding of AK needs.

•**Actual questions about AK:** In Chapter 7 we built another ontology based on architectural review questions that were later answered in an experiment. The ontology provided mostly fitting AK organisation for the review questions.

Above findings suggest that it is important to acquire an accurate understanding of AK needs, as it results in more or less fitting AK organisation. Future work may regard the AK needs of document users as being similar to the requirements for a software system that has to be built. Requirements engineering is a relatively mature discipline from which we can adopt theories, methods, and techniques to acquire an accurate understanding of users' AK needs. An iterative approach can help to refine the understanding of AK needs and detect evolving AK needs.

Above findings also show that the use of different representations of AK needs affects the accuracy of AK needs understanding. We argued that typical questions closely represent the AK needs of users, and we could build a largely fitting ontology-based AK organisation using typical questions. In [21] Clements *et al.* suggest that it is useful for SA document writers to know what questions are being answered by each document section. This suggests that the use of questions to build AK organisations is valuable for future research and practice.

8.3.3 Costs and Benefits of Ontology-based Documentation

A cost-benefit estimation indicated that the adoption of ontology-based SA documentation has a positive return on investment when it replaces file-based SA documentation in two software projects. Further reduction of the cost of creating ontology-based documentation makes it a more cost-effective alternative compared to the use of file-based documentation in industry practice.

The time required to apply our 'typical question' approach and create an ontology may be reduced by automation and tool support. For example, by reusing parts of existing ontologies, mining typical questions from meeting notes, and by applying ontology learning algorithms.

Predefined input forms in ArchiMind, e.g., for the commonly documented properties and relationships of decisions, may reduce the cost of creating AK instances and annotations. We implemented (but not did use) a semi-automatic annotation mechanism in ArchiMind that matches phrases in document content to the names of existing AK instances. The use of natural language processing, as applied in [68], may further reduce the cost of semantic annotation.

Rule-based logic can be used to infer new AK instances and relationships between AK in an ontology, as proposed in [44]. For example, relationships between AK can be inferred from symmetric or transitive properties of existing relationships.

Improving the benefits of ontology-based documentation can also make it a more cost-effective alternative. ArchiMind and other semantic wikis support personalisation and social collaboration within teams, and these features can be extended with ontology support, e.g., for creating personalised views on AK. Evaluating the benefits of combining view-based SA descriptions with an ontology-based approach, e.g., as proposed in [99], is promising future work.

Concerns of AK users are often recorded in different chunks of document content spread across the documentation, and this set of relevant document chunks is different for each AK user [57, 94]. ArchiMind stores chunks of document content in HTML, and it may be promising to investigate the recombination of chunks to generate printable linear SA documents for specific topics, queries, tasks, views, and concerns of AK users. Document chunks that are relevant for individual AK user can be identified by, e.g., modelling users' roles, tasks, and concerns in the ontology, or by latent semantic analysis of SA documents [23].

Integration of ontology-based documentation in the software development process, e.g., in the tools of developers, may improve access to and retrieval of relevant AK. Ontologies are machine interpretable, and tools may be developed to automatically generate design diagrams and source code from ontology-based documentation. Queries and rule-based logic can be used to check for the soundness, correctness, and quality of an SA, its documentation, and other knowledge, e.g., by checking whether every architecturally significant requirement has at least one '*satisfied by*' or '*realized by*' relationship with an AK instance in the ontology.

8.3.4 Generalization of Findings

Replication of our research can potentially help to improve the efficiency and effectiveness of knowledge retrieval in many domains. Our evaluation is specific to the SA domain, as it involves technical SA documentation and experienced software professionals. However, the approaches and solutions that were evaluated are to a large extent generic. We discuss three aspects of our research that may be replicated and generalized to other domains.

Firstly, the identified search strategies, use of prior knowledge, and cognitive biases may apply to document users outside the field of software development. These findings do not seem specific to the software industry domain, other than the use of technical SA documentation and the prior knowledge about software

CHAPTER 8. CONCLUSIONS

systems. It can be worthwhile to replicate this research in other domains to identify and possibly improve document search practices.

Secondly, the ontology-based documentation approach that we proposed is generic. It can use other ontologies and store other types of documentation. The Archi-Mind semantic wiki is an adaptation of OntoWiki, which was designed as a general-purpose semantic wiki for collaborative knowledge engineering [6].

Finally, our ontology engineering approach is largely generic. The approach uses typical questions that documentation users ask, and these users can work in any field. The ontologies created by the approach may be used for other purposes besides the organisation and retrieval of documented AK.

9

Samenvatting

Software heeft vooruitgang in veel vakgebieden mogelijk gemaakt en heeft een toenemend invloed op ons leven en de samenleving in zijn geheel. Software wordt gebruikt in computers, communicatienetwerken, medische apparatuur, fabrieken, vliegtuigen, treinen, auto's, mobiele telefoons, huishoudapparatuur, enzovoort. Software systemen die verkeerd ontworpen, gebouwd, of onderhouden zijn kunnen slecht functioneren en daardoor traag, onveilig, of onbetrouwbaar worden, wat resulteert in het verlies van informatie, tijd, geld, of levens.

Het is belangrijk om goed functionerende software te ontwikkelen. Dit is echter niet eenvoudig; het ontwikkelen van een software systeem kan jaren werk door honderden professionals en miljoenen regels code vereisen. In dit proefschrift doen we onderzoek naar het verbeteren van de kennisvergaring uit de documentatie die professionals gebruiken tijdens software ontwikkeling.

Tijdens het ontwikkelen van een groot software systeem werken meerdere professionals samen in een project. Projectmatige softwareontwikkeling wordt gepland in iteratieve cycli en fasen, bijvoorbeeld in een requirement, ontwerp, en implementatiefase. In software projecten worden diverse documentatietypes gebruikt, zoals documenten voor systeemontwerp en documenten voor de functionele vereisten aan een systeem. Deze software documentatie is essentieel voor de communicatie van kennis tussen professionals, vooral als ze werken in verschillende projectfasen, vestigingen, en tijdzones.

Eén van de eerste activiteiten in een software project is het specificeren van de Software Architectuur (SA) van een systeem. In een SA ontwerp wordt het systeem gedecomposeerd in componenten die met elkaar communiceren d.m.v. interacties. Een SA ontwerp realiseert functionele en non-functionele vereisten aan een systeem (bijvoorbeeld snelheid en betrouwbaarheid), randvoorwaarden

vanuit de technische en organisatorische omgeving, werk-verdeling, budget, planning, en hergebruik van componenten.

Het documenteren van SA dient drie doelen: het wordt gebruikt voor systeem-analyse, voor onderwijs, en als de primaire vorm van communicatie tussen de belanghebbenden in een software project. SA documentatie speelt niet alleen een rol in het begin van een project, maar ook later tijdens onderhoud en verbetering van een systeem. SA documentatie bevat Architectuur Kennis (AK). AK kan worden omschreven als: "*De geïntegreerde representatie van de software architectuur van een software-intensief systeem (of familie van systemen), de beslissingen over het architectuurontwerp, en de externe context/omgeving*".

In het bedrijfsleven is het gebruikelijk om AK op te slaan in bestand-gebaseerde documenten, zoals tekst- en diagrambestanden. Het opdelen van de inhoud van documenten in secties en subsecties zorgt voor een organisatie van de AK in de documenten. Een inhoudsopgave met sectietitels kan worden gebruikt als een index tijdens het zoeken naar AK in deze organisatie.

In SA documentatie worden veel relaties tussen AK beschreven, bijvoorbeeld tussen beslissingen, componenten, en functionele vereisten. Zo kan een enkele beslissing al veel relaties hebben met andere AK. Een ontwikkelaar kan zich afvragen wat de impact is van deze beslissing op de componenten en de interfaces waar hij of zij verantwoordelijk voor is. Een architect die de beslissing evalueert heeft interesse in gerelateerde (alternatieve) beslissing en vereisten. Een kwaliteitsbewaker kan zich afvragen welke kwaliteitsattributen door de beslissing veranderen.

Als AK en de relaties tussen AK niet zijn geïndexeerd in een documentorganisatie moet de AK worden gezocht in de documentinhoud binnenin secties. Het doorlezen of met sleutelwoorden doorzoeken van documentinhoud kan echter veel tijd kosten en foutgevoelig zijn wegens synoniemen, spelfouten, en afkorting.

Het is moeilijk om intergerelateerde AK te organiseren in de lineaire inhoudsopgave van bestand-gebaseerde documenten op een manier dat alle documentgebruikers worden ondersteund in het vinden van de AK waar ze behoefte aan hebben. SA documenten hebben vaak een uniforme '*één maat past iedereen*' organisatie die weinig ondersteuning biedt voor de taken van individuele gebruikers van AK.

Een organisatie van AK die weinig ondersteuning biedt voor de behoefte aan AK kan er voor zorgen dat documentiegebruikers incorrecte en incomplete AK vinden of tijd verspillen omdat ze op de verkeerde plek naar AK zoeken. Het inefficiënt en ineffectief verkrijgen van AK uit SA documentatie is het probleem waar dit proefschrift zich op richt.

We vermoeden dat het gebruik van *ontologie-gebaseerde* SA documentatie het verkrijgen van AK kan verbeteren in vergelijking met het gebruik van bestands-gebaseerde documentatie. "*Een ontologie*" refereert aan een formeel domeinmodel waarin concepten en relaties worden beschreven. Een ontologie kan de AK in SA documentatie organiseren m.b.v. klassen en relaties, en kan de betekenis (semantiek) van AK expliciet maken waardoor documentatiegebruikers de AK en relaties tussen AK kunnen herkennen. Een ontologie-gebaseerde AK organisatie is niet lineair, wat documentatiegebruikers kan helpen om de intergerelateerde AK die ze nodig hebben snel en correct te verkrijgen.

De hoofdvraag die we onderzoeken in dit proefschrift is of het gebruik van ontologie-gebaseerde documentatie de efficiëntie en effectiviteit van het verkrijgen van AK kan verbeteren. We bestuderen efficiëntie door het meten van de tijd die benodigd is om antwoord te geven op vragen die gaan over AK, en bestuderen effectiviteit door het meten van de correctheid en compleetheid van antwoorden.

We hebben eerst onderzocht hoe professionals in het bedrijfsleven AK verkrijgen uit bestands-gebaseerde documentatie, om zo de huidige praktijk te begrijpen en eventuele verbeteringen te identificeren. We hebben het zoekgedrag van professionals bestudeerd en konden zo vier zoekstrategieën identificeren. Verder kwamen we erachter dat professionals onzekerheid en uitdagingen ervaren wanneer ze tijdens het zoeken niet in staat zijn om een documentorganisatie te volgen die gereleerd is aan de vraag die ze beantwoorden. Het gebruik van voorkennis helpt om met onzekerheden om te gaan, maar leidt ook vaak tot vooringenomenheid, inefficiëntie, en ineffectiviteit tijdens het zoeken naar AK.

Met behulp van literatuuronderzoek identificeerden we een aantal achterliggende oorzaken van de uitdagingen tijdens het verkrijgen van AK uit bestands-gebaseerde documentatie. Uitdagingen tijdens het verkrijgen van AK in het bedrijfsleven kunnen deels worden herleid naar de lineaire organisatie van AK in bestands-gebaseerde documenten. Dit suggereert dat er in de praktijk ruimte is voor het verbeteren van de efficiëntie en effectiviteit van het verkrijgen van AK.

Verder hebben we onderzocht hoe een ontologie kan worden gebruikt voor het verkrijgen van AK uit documentatie. We introduceerden een ontologie-gebaseerde documentatiemethode die gebruik maakt van een software ontologie en een semantische wiki voor het organiseren en opzoeken van AK. Met de semantische wiki kan de inhoud van bestands-gebaseerde documentatie worden opgeslagen in webpagina's die men kan annoteren, doorzoeken, en navigeren m.b.v. een ontologie. Uitdagingen tijdens het verkrijgen van AK uit bestands-gebaseerde documentatie kunnen deels worden verholpen door het gebruik van ontologie-gebaseerde documentatie.

Vervolgens onderzochten we hoe een ontologie voor SA documentatie gebouwd kan worden in de context van een software project. We introduceerden een methode voor het bouwen van ontologieën die gebruikt maakt van typische vragen van software professionals over AK. Met een verkennende casestudy hebben we geëvalueerd hoe goed de methode werkt om een ontologie te bouwen in een software project met diverse gebruikers, domein complexiteit, en andere contextuele factoren.

Om zekerheid te krijgen dat ontologie-gebaseerde documentatie het verkrijgen van AK kan verbeteren hebben we deze vergeleken met bestands-gebaseerde documentatie. We hebben een experiment uitgevoerd in twee bedrijven waarin software professionals vragen over AK beantwoorden, om zo te onderzoeken hoe het gebruik van bestands-gebaseerde en ontologie-gebaseerde documentatie de efficiëntie en effectiviteit van het verkrijgen van AK beïnvloed. Het gebruik van ontologie-gebaseerde documentatie was significant meer efficiënt en effectief dan het gebruik van bestands-gebaseerde documentatie.

Een deel van beschikbare AK organisatie in het experiment was passend voor het verkrijgen van AK; de organisatie had een expliciete beschrijving van de types AK en relaties tussen AK die ook in de vragen in het experiment waren beschreven. Door analyse van de zoekacties van professionals kwamen we erachter dat het gebruik van passende AK organisatie een positieve correlatie had met de efficiëntie en effectiviteit van het verkrijgen van AK uit bestands-gebaseerde en ontologie-gebaseerde documentatie. De correlatie geeft een verklaring voor het verschil in efficiëntie en effectiviteit tussen de twee vormen van documentatie.

Deelnemers in het experiment gaven via een vragenlijst een grotendeels positieve evaluatie van ontologie-gebaseerde documentatie. Een ruwe kosten-baten schatting wijst op een positief netto resultaat wanneer de bestands-gebaseerde documentatie zou worden vervangen door ontologie-gebaseerde documentatie in de bestudeerde software projecten.

Om erachter te komen hoe verschillende ontologieën presteren t.o.v. elkaar hebben we onderzocht hoe het gebruik van verschillende ontologie-gebaseerde AK organisaties invloed heeft op de efficiëntie en effectiviteit van het verkrijgen van AK. Door middel van een experiment hebben we onderzocht wat het verschil is tussen het verkrijgen van AK m.b.v. een ontologie die gebouwd is op basis van de verwachte behoefte aan AK van documentatiegebruikers, en het verkrijgen van AK m.b.v. een ontologie die is gebouwd op basis van de daadwerkelijke behoefte aan AK. De resultaten laten zien dat een beter begrip van de behoefte aan AK kan worden gebruikt om ontologieën te bouwen met meer passende AK organisatie, waardoor de efficiëntie en effectiviteit van het verkrijgen van AK verbeterd.

De bevindingen in dit proefschrift geven een bevestigend antwoord op de hoofdvraag: "*Kunnen we de efficiëntie en effectiviteit van het verkrijgen van AK verbeteren door het gebruik van ontologie-gebaseerde documentatie?*".

We beschrijven de bevindingen in dezelfde terminologie als de hoofdvraag, om zo het antwoord op de hoofdvraag samen te vatten.

- De efficiëntie en effectiviteit van het verkrijgen van AK uit bestands-gebaseerde documentatie kan worden verbeterd.
- Gebruik van ontologie-gebaseerde documentatie kan uitdagingen verhelpen die bijdragen aan het inefficiënt en ineffectief verkrijgen van AK uit bestands-gebaseerde documentatie.
- Een nuttige ontologie voor het verkrijgen van AK uit ontologie-gebaseerde documentatie kan worden gebouwd in een software project.
- Het gebruik van ontologie-gebaseerde documentatie kan de efficiëntie en effectiviteit van het verkrijgen van AK verbeteren in vergelijking met het gebruik van bestands-gebaseerde documentatie.
- De efficiëntie en effectiviteit van het verkrijgen van AK in ontologie-gebaseerde documentatie kan worden verbeterd door het gebruik van ontologie-gebaseerde AK organisatie die is gebouwd op basis van een beter begrip van de behoefte aan AK.

De bevindingen laten zien dat het gebruik van ontologie-gebaseerde documentatie veelbelovend is. Tevens laten de bevindingen zien hoe men verbeteringen kan aanbrengen in bestands-gebaseerde documentatie die momenteel veel wordt gebruikt in het bedrijfsleven. Er is echter meer onderzoek nodig om de bevindingen verder te staven en generaliseren naar verschillende typen software projecten, en om open vragen en toekomstig werk te verkennen.

SIKS Dissertatiereeks

=====
1998
=====

- 1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects
- 1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information
- 1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the
Language/Action Perspective
- 1998-4 Dennis Breuker (UM)
Memory versus Search in Games
- 1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

=====
1999
=====

- 1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of
Agricultural Products
- 1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets
- 1999-3 Don Beal (UM)
The Nature of Minimax Search
- 1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects
- 1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of
Network Information Systems
- 1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems
- 1999-7 David Spelt (UT)
Verification support for object database design
- 1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism
for Discrete Reallocation.

=====
2000
=====

- 2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance
- 2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management
- 2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering
en actorperspectief.
- 2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design
- 2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.
- 2000-6 Rogier van Eijk (UU)

- Programming Languages for Agent Communication
- 2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management
- 2000-8 Veerle Coupéf (EUR)
Sensitivity Analysis of Decision-Theoretic Networks
- 2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization
- 2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture
- 2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

=====
2001
=====

- 2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks
- 2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models
- 2001-3 Maarten van Someren (UvA)
Learning as problem solving
- 2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
- 2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style
- 2001-6 Martijn van Welie (VU)
Task-based User Interface Design
- 2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization
- 2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
- 2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views
of Packages as Classes
- 2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice. BRAHMS: a multiagent modeling
and simulation language for work practice analysis and design
- 2001-11 Tom M. van Engers (VUA)
Knowledge Management: The Role of Mental Models in Business Systems Design

=====
2002
=====

- 2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis
- 2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections
- 2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval
- 2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining
- 2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by
Privacy-concerned Agents
- 2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the
legal domain
- 2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

- 2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
- 2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems
- 2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble
- 2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications
- 2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems
- 2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications
- 2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying
Multi-Agent Systems
- 2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling
- 2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications
- 2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

=====
2003
=====

- 2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments
- 2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems
- 2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
- 2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology
- 2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach
- 2003-06 Boris van Schooten (UT)
Development and specification of virtual environments
- 2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks
- 2003-08 Yongping Ran (UM)
Repair Based Scheduling
- 2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour
- 2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between
medium, innovation context and culture
- 2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
- 2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval
- 2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models
- 2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
- 2003-15 Mathijs de Weerd (TUD)
Plan Merging in Multi-Agent Systems
- 2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media
Warehouses
- 2003-17 David Jansen (UT)

Extensions of Statecharts with Probability, Time, and Stochastic Timing
2003-18 Levente Kocsis (UM)
Learning Search Decisions

=====
2004
=====

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic
2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business
2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures
2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity
2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques
2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar
abstract denken, vooral voor meisjes
2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale
politiële gegevensuitwisseling en digitale expertise
2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning
2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects
2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies
2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents
2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play
2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium
2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining
2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning
2004-17 Mark Winands (UM)
Informed Search in Complex Games
2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models
2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval
2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

=====
2005
=====

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications
2005-02 Erik van der Werf (UM)
AI techniques for the game of Go
2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

- 2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data
- 2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing
- 2005-06 Pieter Spronck (UM)
Adaptive Game AI
- 2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems
- 2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications
- 2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages
- 2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
- 2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
- 2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry
- 2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
- 2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
- 2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes
- 2005-16 Joris Graaumans (UU)
Usability of XML Query Languages
- 2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components
- 2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks
- 2005-19 Michel van Dartel (UM)
Situated Representation
- 2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives
- 2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

====
2006
====

- 2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting
- 2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations
- 2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems
- 2006-04 Marta Sabou (VU)
Building Web Service Ontologies
- 2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines
- 2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
- 2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering
- 2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web

- 2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion
- 2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems
- 2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types
- 2006-12 Bert Bongers (VU)
Interactivation - Towards an e-ecology of people, our technological environment, and the arts
- 2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents
- 2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
- 2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain
- 2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks
- 2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device
- 2006-18 Valentin Zhizhkhun (UVA)
Graph transformation for Natural Language Processing
- 2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach
- 2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining
- 2006-21 Bas van Gils (RUN)
Aptness on the Web
- 2006-22 Paul de Vrieze (RUN)
Fundamentals of Adaptive Personalisation
- 2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web
- 2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources
- 2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC
- 2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval
- 2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories
- 2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

=====
2007
=====

- 2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures
- 2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach
- 2007-03 Peter Mika (VU)
Social Networks and the Semantic Web
- 2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach
- 2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

- 2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs
- 2007-07 Natasa Jovanovic' (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings
- 2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations
- 2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation
- 2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols
- 2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System
- 2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty
- 2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology
- 2007-14 Niek Bergboer (UM)
Context-Based Image Analysis
- 2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model
- 2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems
- 2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice
- 2007-18 Bart Orriens (UvT)
On the development an management of adaptive business collaborations
- 2007-19 David Levy (UM)
Intimate relationships with artificial partners
- 2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network
- 2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005
- 2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns
- 2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems
- 2007-24 Georgina Ramirez Camps (CWI)
Structural Features in XML Retrieval
- 2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement

=====
2008
=====

- 2008-01 Katalin Boer-Sorban (EUR)
Agent-Based Simulation of Financial Markets: A modular, continuous-time approach
- 2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations
- 2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: a usage-based approach
- 2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration
- 2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective
- 2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines, an

Artificial Intelligence Perspective

- 2008-07 Peter van Rosmalen (OU)
Supporting the tutor in the design and support of adaptive e-learning
- 2008-08 Janneke Bolt (UU)
Bayesian Networks: Aspects of Approximate Inference
- 2008-09 Christof van Nimwegen (UU)
The paradox of the guided user: assistance can be counter-effective
- 2008-10 Wauter Bosma (UT)
Discourse oriented summarization
- 2008-11 Vera Kartseva (VU)
Designing Controls for Network Organizations: A Value-Based Approach
- 2008-12 Jozsef Farkas (RUN)
A Semiotically Oriented Cognitive Model of Knowledge Representation
- 2008-13 Caterina Carraciolo (UVA)
Topic Driven Access to Scientific Handbooks
- 2008-14 Arthur van Bunningen (UT)
Context-Aware Querying; Better Answers with Less Effort
- 2008-15 Martijn van Otterlo (UT)
The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.
- 2008-16 Henriette van Vugt (VU)
Embodied agents from a user's perspective
- 2008-17 Martin Op 't Land (TUD)
Applying Architecture and Ontology to the Splitting and Allying of Enterprises
- 2008-18 Guido de Croon (UM)
Adaptive Active Vision
- 2008-19 Henning Rode (UT)
From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search
- 2008-20 Rex Arendsen (UVA)
Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven.
- 2008-21 Krisztian Balog (UVA)
People Search in the Enterprise
- 2008-22 Henk Koning (UU)
Communication of IT-Architecture
- 2008-23 Stefan Visscher (UU)
Bayesian network models for the management of ventilator-associated pneumonia
- 2008-24 Zharko Aleksovski (VU)
Using background knowledge in ontology matching
- 2008-25 Geert Jonker (UU)
Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency
- 2008-26 Marijn Huijbregts (UT)
Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled
- 2008-27 Hubert Vogten (OU)
Design and Implementation Strategies for IMS Learning Design
- 2008-28 Ildiko Flesch (RUN)
On the Use of Independence Relations in Bayesian Networks
- 2008-29 Dennis Reidsma (UT)
Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans
- 2008-30 Wouter van Atteveldt (VU)
Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content
- 2008-31 Loes Braun (UM)
Pro-Active Medical Information Retrieval
- 2008-32 Trung H. Bui (UT)
Toward Affective Dialogue Management using Partially Observable Markov Decision Processes

- 2008-33 Frank Terpstra (UVA)
Scientific Workflow Design; theoretical and practical issues
- 2008-34 Jeroen de Knijff (UU)
Studies in Frequent Tree Mining
- 2008-35 Ben Torben Nielsen (UvT)
Dendritic morphologies: function shapes structure

=====
2009
=====

- 2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models
- 2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques
- 2009-03 Hans Stol (UvT)
A Framework for Evidence-based Policy Making Using IT
- 2009-04 Josephine Nabukenya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering
- 2009-05 Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality
- 2009-06 Muhammad Subianto (UU)
Understanding Classification
- 2009-07 Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion
- 2009-08 Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments
- 2009-09 Benjamin Kanagwa (RUN)
Design, Discovery and Construction of Service-oriented Systems
- 2009-10 Jan Wielemaker (UVA)
Logic programming for knowledge-intensive interactive applications
- 2009-11 Alexander Boer (UVA)
Legal Theory, Sources of Law & the Semantic Web
- 2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)
Operating Guidelines for Services
- 2009-13 Steven de Jong (UM)
Fairness in Multi-Agent Systems
- 2009-14 Maksym Korotkiy (VU)
From ontology-enabled services to service-enabled ontologies
(making ontologies work in e-science with ONTO-SOA)
- 2009-15 Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense
- 2009-16 Fritz Reul (UvT)
New Architectures in Computer Chess
- 2009-17 Laurens van der Maaten (UvT)
Feature Extraction from Visual Data
- 2009-18 Fabian Groffen (CWI)
Armada, An Evolving Database System
- 2009-19 Valentin Robu (CWI)
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
- 2009-20 Bob van der Vecht (UU)
Adjustable Autonomy: Controlling Influences on Decision Making
- 2009-21 Stijn Vanderlooy (UM)
Ranking and Reliable Classification
- 2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence
- 2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment
- 2009-24 Annerieke Heuvelink (VUA)

- Cognitive Models for Training Simulations
- 2009-25 Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"
- 2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services
- 2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web
- 2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models
- 2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications
- 2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage
- 2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text
- 2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors
- 2009-33 Khiet Truong (UT)
How Does Real Affect Affect Recognition In Speech?
- 2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method
Engineering Approach
- 2009-35 Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling
- 2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks
- 2009-37 Hendrik Drachslar (OUN)
Navigation Support for Learners in Informal Learning Networks
- 2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning
resources in a multilingual context
- 2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets
- 2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language
- 2009-41 Igor Berezhnyy (UvT)
Digital Analysis of Paintings
- 2009-42 Toine Bogers (UvT)
Recommender Systems for Social Bookmarking
- 2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic
Search and Mobile Ambients
- 2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations
- 2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful
- 2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

=====
2010
=====

- 2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter
- 2010-02 Ingo Wassink (UT)
Work flows in Life Science
- 2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents
- 2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of

- Co-located Teams in Multidisplay Environments
- 2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems
- 2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI
- 2010-07 Wim Fikkert (UT)
Gesture interaction at a Distance
- 2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software.
Protecting user freedoms in a world of software communities and eGovernments
- 2010-09 Hugo Kielman (UL)
A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging
- 2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children
- 2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning
- 2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis
- 2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques
- 2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration
- 2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models
- 2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice
- 2010-17 Spyros Koutoulas (VU)
Scalable Discovery of Networked Resources: Algorithms,
Infrastructure, Applications
- 2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation
- 2010-19 Henriette Cramer (UvA)
People's Responses to Autonomous and Adaptive Systems
- 2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and
Authorship of Emergent Narrative
- 2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation
- 2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data
- 2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions
- 2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies
- 2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective
- 2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines
- 2010-27 Marten Voulon (UL)
Automatisch contracteren
- 2010-28 Arne Koopman (UU)
Characteristic Relational Patterns
- 2010-29 Stratos Idreos (CWI)
Database Cracking: Towards Auto-tuning Database Kernels
- 2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning,
structuring, and retrieval
- 2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web
- 2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically

- solving Interoperability Problems
- 2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval
- 2010-34 Teduh Dirgahayu (UT)
Interaction Design in Service Compositions
- 2010-35 Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval
- 2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification
- 2010-37 Niels Lohmann (TUE)
Correctness of services and their composition
- 2010-38 Dirk Fahland (TUE)
From Scenarios to components
- 2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents
- 2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web
- 2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search
- 2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach
- 2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies
- 2010-44 Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain
- 2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services
- 2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment
- 2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples
- 2010-48 Withdrawn
- 2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions
- 2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives
- 2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources
- 2010-52 Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention
- 2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access
- ====
- 2011
- ====
- 2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models
- 2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language
- 2011-03 Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems
- 2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

- 2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.
- 2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage
- 2011-07 Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction
- 2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues
- 2011-09 Tim de Jong (OU)
Contextualised Mobile Media for Learning
- 2011-10 Bart Bogaert (UvT)
Cloud Content Contention
- 2011-11 Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective
- 2011-12 Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining
- 2011-13 Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling
- 2011-14 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets
- 2011-15 Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval
- 2011-16 Maarten Schadd (UM)
Selective Search in Games of Different Complexity
- 2011-17 Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness
- 2011-18 Mark Ponsen (UM)
Strategic Decision-Making in complex games
- 2011-19 Ellen Rusman (OU)
The Mind's Eye on Personal Profiles
- 2011-20 Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach
- 2011-21 Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems
- 2011-22 Junte Zhang (UVA)
System Evaluation of Archival Description and Access
- 2011-23 Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media
- 2011-24 Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination with Virtual Humans
On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
- 2011-25 Syed Waqar ul Qounain Jaffry (VU)
Analysis and Validation of Models for Trust Dynamics
- 2011-26 Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance
Trade-Offs in Embodied Conversational Agents and Robots
- 2011-27 Aniel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns
- 2011-28 Rianne Kaptein(UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure
- 2011-29 Faisal Kamiran (TUE)
Discrimination-aware Classification
- 2011-30 Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions
- 2011-31 Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
- 2011-32 Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science
- 2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

- 2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
- 2011-35 Maaïke Harbers (UU)
Explaining Agent Behavior in Virtual Training
- 2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach
- 2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning
and Supervised Network Inference
- 2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization
- 2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games
- 2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development
- 2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control
- 2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution
- 2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge
- 2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces
- 2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection
- 2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture
for the Domain of Mobile Police Work
- 2011-47 Azizi Bin Ab Aziz(VU)
Exploring Computational Models for Intelligent Support of Persons with Depression
- 2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
- 2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues:
design aspects influencing interaction quality

=====
2012
=====

- 2012-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda
- 2012-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
- 2012-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories
- 2012-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications
- 2012-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems
- 2012-06 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks
- 2012-07 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human
Performance under Demanding Conditions
- 2012-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories
- 2012-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms
- 2012-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment

- 2012-11 J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
- 2012-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems
- 2012-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
- 2012-14 Evgeny Knutov (TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems
- 2012-15 Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.
- 2012-16 Fiemke Both (VU)
Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
- 2012-17 Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance
- 2012-18 Eltjo Poort (VU)
Improving Solution Architecting Practices
- 2012-19 Helen Schonenberg (TUE)
What's Next? Operational Support for Business Process Execution
- 2012-20 Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
- 2012-21 Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval
- 2012-22 Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
- 2012-23 Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
- 2012-24 Laurens van der Werff (UT)
Evaluation of Noisy Transcripts for Spoken Document Retrieval
- 2012-25 Silja Eckartz (UT)
Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
- 2012-26 Emile de Maat (UVA)
Making Sense of Legal Text
- 2012-27 Hayrettin Gurkok (UT)
Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
- 2012-28 Nancy Pascall (UvT)
Engendering Technology Empowering Women
- 2012-29 Almer Tigelaar (UT)
Peer-to-Peer Information Retrieval
- 2012-30 Alina Pommeranz (TUD)
Designing Human-Centered Systems for Reflective Decision Making
- 2012-31 Emily Bagarukayo (RUN)
A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
- 2012-32 Wietske Visser (TUD)
Qualitative multi-criteria preference representation and reasoning
- 2012-33 Rory Sie (OUN)
Coalitions in Cooperation Networks (COCOON)
- 2012-34 Pavol Jancura (RUN)
Evolutionary analysis in PPI networks and applications
- 2012-35 Evert Haasdijk (VU)
Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics
- 2012-36 Denis Ssebugwawo (RUN)
Analysis and Evaluation of Collaborative Modeling Processes
- 2012-37 Agnes Nakakawa (RUN)
A Collaboration Process for Enterprise Architecture Creation
- 2012-38 Selmar Smit (VU)
Parameter Tuning and Scientific Testing in Evolutionary Algorithms

- 2012-39 Hassan Fatemi (UT)
Risk-aware design of value and coordination networks
- 2012-40 Agus Gunawan (UvT)
Information Access for SMEs in Indonesia
- 2012-41 Sebastian Kelle (OU)
Game Design Patterns for Learning
- 2012-42 Dominique Verpoorten (OU)
Reflection Amplifiers in self-regulated Learning
- 2012-43 Withdrawn
- 2012-44 Anna Tordai (VU)
On Combining Alignment Techniques
- 2012-45 Benedikt Kratz (UvT)
A Model and Language for Business-aware Transactions
- 2012-46 Simon Carter (UVA)
Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
- 2012-47 Manos Tsagkias (UVA)
Mining Social Media: Tracking Content and Predicting Behavior
- 2012-48 Jorn Bakker (TUE)
Handling Abrupt Changes in Evolving Time-series Data
- 2012-49 Michael Kaisers (UM)
Learning against Learning - Evolutionary dynamics of reinforcement
learning algorithms in strategic interactions
- 2012-50 Steven van Kervel (TUD)
Ontology driven Enterprise Information Systems Engineering
- 2012-51 Jeroen de Jong (TUD)
Heuristics in Dynamic Sceduling; a practical framework with a case
study in elevator dispatching

=====
2013
=====

- 2013-01 Viorel Milea (EUR)
News Analytics for Financial Decision Support
- 2013-02 Erietta Liarou (CWI)
MonetDB/DataCell: Leveraging the Column-store Database Technology
for Efficient and Scalable Stream Processing
- 2013-03 Szymon Klarman (VU)
Reasoning with Contexts in Description Logics
- 2013-04 Chetan Yadati(TUD)
Coordinating autonomous planning and scheduling
- 2013-05 Dulce Pumareja (UT)
Groupware Requirements Evolutions Patterns
- 2013-06 Romulo Goncalves(CWI)
The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
- 2013-07 Giel van Lankveld (UvT)
Quantifying Individual Player Differences
- 2013-08 Robbert-Jan Merk(VU)
Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
- 2013-09 Fabio Gori (RUN)
Metagenomic Data Analysis: Computational Methods and Applications
- 2013-10 Jeewanie Jayasinghe Arachchige(UvT)
A Unified Modeling Framework for Service Design.
- 2013-11 Evangelos Pournaras(TUD)
Multi-level Reconfigurable Self-organization in Overlay Services
- 2013-12 Marian Razavian(VU)
Knowledge-driven Migration to Services
- 2013-13 Mohammad Safiri(UT)
Service Tailoring: User-centric creation of integrated IT-based
homecare services to support independent living of elderly
- 2013-14 Jafar Tanha (UVA)

- Ensemble Approaches to Semi-Supervised Learning Learning
- 2013-15 Daniel Hennes (UM)
Multiagent Learning - Dynamic Games and Applications
- 2013-16 Eric Kok (UU)
Exploring the practical benefits of argumentation in multi-agent deliberation
- 2013-17 Koen Kok (VU)
The PowerMatcher: Smart Coordination for the Smart Electricity Grid
- 2013-18 Jeroen Janssens (UvT)
Outlier Selection and One-Class Classification
- 2013-19 Renze Steenhuizen (TUD)
Coordinated Multi-Agent Planning and Scheduling
- 2013-20 Katja Hofmann (UvA)
Fast and Reliable Online Learning to Rank for Information Retrieval
- 2013-21 Sander Wubben (UvT)
Text-to-text generation by monolingual machine translation
- 2013-22 Tom Claassen (RUN)
Causal Discovery and Logic
- 2013-23 Patricio de Alencar Silva(UvT)
Value Activity Monitoring
- 2013-24 Haitham Bou Ammar (UM)
Automated Transfer in Reinforcement Learning
- 2013-25 Agnieszka Anna Latoszek-Berendsen (UM)
Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
- 2013-26 Alireza Zarghami (UT)
Architectural Support for Dynamic Homecare Service Provisioning
- 2013-27 Mohammad Huq (UT)
Inference-based Framework Managing Data Provenance
- 2013-28 Frans van der Sluis (UT)
When Complexity becomes Interesting: An Inquiry into the Information eXperience
- 2013-29 Iwan de Kok (UT)
Listening Heads
- 2013-30 Joyce Nakatumba (TUE)
Resource-Aware Business Process Management: Analysis and Support
- 2013-31 Dinh Khoa Nguyen (UvT)
Blueprint Model and Language for Engineering Cloud Applications
- 2013-32 Kamakshi Rajagopal (OUN)
Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development
- 2013-33 Qi Gao (TUD)
User Modeling and Personalization in the Microblogging Sphere
- 2013-34 Kien Tjin-Kam-Jet (UT)
Distributed Deep Web Search
- 2013-35 Abdallah El Ali (UvA)
Minimal Mobile Human Computer Interaction
- 2013-36 Than Lam Hoang (TUE)
Pattern Mining in Data Streams
- 2013-37 Dirk Börner (OUN)
Ambient Learning Displays
- 2013-38 Eelco den Heijer (VU)
Autonomous Evolutionary Art
- 2013-39 Joop de Jong (TUD)
A Method for Enterprise Ontology based Design of Enterprise Information Systems
- 2013-40 Pim Nijssen (UM)
Monte-Carlo Tree Search for Multi-Player Games
- 2013-41 Jochem Liem (UVA)
Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
- 2013-42 Leon Planken (TUD)
Algorithms for Simple Temporal Reasoning
- 2013-43 Marc Bron (UVA)

Exploration and Contextualization through Interaction and Concepts

=====
2014
=====

- 2014-01 Nicola Barile (UU)
Studies in Learning Monotone Models from Data
- 2014-02 Fiona Tuliayo (RUN)
Combining System Dynamics with a Domain Modeling Method
- 2014-03 Sergio Raul Duarte Torres (UT)
Information Retrieval for Children: Search Behavior and Solutions
- 2014-04 Hanna Jochmann-Mannak (UT)
Websites for children: search strategies and interface design -
Three studies on children's search performance and evaluation
- 2014-05 Jurriaan van Reijssen (UU)
Knowledge Perspectives on Advancing Dynamic Capability
- 2014-06 Damian Tamburri (VU)
Supporting Networked Software Development
- 2014-07 Arya Adriansyah (TUE)
Aligning Observed and Modeled Behavior
- 2014-08 Samur Araujo (TUD)
Data Integration over Distributed and Heterogeneous Data Endpoints
- 2014-09 Philip Jackson (UvT)
Toward Human-Level Artificial Intelligence: Representation and
Computation of Meaning in Natural Language
- 2014-10 Ivan Salvador Razo Zapata (VU)
Service Value Networks
- 2014-11 Janneke van der Zwaan (TUD)
An Empathic Virtual Buddy for Social Support
- 2014-12 Willem van Willigen (VU)
Look Ma, No Hands: Aspects of Autonomous Vehicle Control
- 2014-13 Arlette van Wissen (VU)
Agent-Based Support for Behavior Change: Models and Applications in
Health and Safety Domains
- 2014-14 Yangyang Shi (TUD)
Language Models With Meta-information
- 2014-15 Natalya Mogles (VU)
Agent-Based Analysis and Support of Human Functioning in
Complex Socio-Technical Systems: Applications in Safety and Healthcare
- 2014-16 Krystyna Milian (VU)
Supporting trial recruitment and design by automatically
interpreting eligibility criteria
- 2014-17 Kathrin Dentler (VU)
Computing healthcare quality indicators automatically: Secondary Use
of Patient Data and Semantic Interoperability
- 2014-18 Matijts Ghijsen (VU)
Methods and Models for the Design and Study of Dynamic Agent Organizations
- 2014-19 Vincius Ramos (TUE)
Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
- 2014-20 Mena Habib (UT)
Named Entity Extraction and Disambiguation for Informal Text: The Missing Link
- 2014-21 Cassidy Clark (TUD)
Negotiation and Monitoring in Open Environments
- 2014-22 Marieke Peeters (UU)
Personalized Educational Games - Developing agent-supported scenario-based training
- 2014-23 Eleftherios Sidirourgos (UvA/CWI)
Space Efficient Indexes for the Big Data Era
- 2014-24 Davide Ceolin (VU)
Trusting Semi-structured Web Data
- 2014-25 Martijn Lappenschaar (RUN)

- New network models for the analysis of disease interaction
- 2014-26 Tim Baarslag (TUD)
What to Bid and When to Stop
- 2014-27 Rui Jorge Almeida (EUR)
Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
- 2014-28 Anna Chmielowiec (VU)
Decentralized k-Clique Matching
- 2014-29 Jaap Kabbedijk (UU)
Variability in Multi-Tenant Enterprise Software
- 2014-30 Peter de Cock (UvT)
Anticipating Criminal Behaviour
- 2014-31 Leo van Moergestel (UU)
Agent Technology in Agile Multiparallel Manufacturing and Product Support
- 2014-32 Naser Ayat (UvA)
On Entity Resolution in Probabilistic Data
- 2014-33 Tesfa Tegegne (RUN)
Service Discovery in eHealth
- 2014-34 Christina Manteli (VU)
The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.
- 2014-35 Joost van Ooijen (UU)
Cognitive Agents in Virtual Worlds: A Middleware Design Approach
- 2014-36 Joos Buijs (TUE)
Flexible Evolutionary Algorithms for Mining Structured Process Models
- 2014-37 Maral Dadvar (UT)
Experts and Machines United Against Cyberbullying
- 2014-38 Danny Plass-Oude Bos (UT)
Making brain-computer interfaces better: improving usability through post-processing.
- 2014-39 Jasmina Maric (UvT)
Web Communities, Immigration, and Social Capital
- 2014-40 Walter Omona (RUN)
A Framework for Knowledge Management Using ICT in Higher Education
- 2014-41 Frederic Hogenboom (EUR)
Automated Detection of Financial Events in News Text
- 2014-42 Carsten Eijckhof (CWI/TUD)
Contextual Multidimensional Relevance Models
- 2014-43 Kevin Vlaanderen (UU)
Supporting Process Improvement using Method Increments
- 2014-44 Paulien Meesters (UvT)
Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.
- 2014-45 Birgit Schmitz (OUN)
Mobile Games for Learning: A Pattern-Based Approach
- 2014-46 Ke Tao (TUD)
Social Web Data Analytics: Relevance, Redundancy, Diversity
- 2014-47 Shangsong Liang (UVA)
Fusion and Diversification in Information Retrieval

====

2015

====

- 2015-01
Niels Netten (UvA)
Machine Learning for Relevance of Information in Crisis Response
- 2015-02 Faiza Bukhsh (UvT)
Smart auditing: Innovative Compliance Checking in Customs Controls
- 2015-03 Twan van Laarhoven (RUN)
Machine learning for network data
- 2015-04 Howard Spoelstra (OUN)
Collaborations in Open Learning Environments

- 2015-05 Christoph Böch(UT)
Cryptographically Enforced Search Pattern Hiding
- 2015-06 Farideh Heidari (TUD)
Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes
- 2015-07 Maria-Hendrike Peetz(UvA)
Time-Aware Online Reputation Analysis
- 2015-08 Jie Jiang (TUD)
Organizational Compliance: An agent-based model for designing and evaluating organizational interactions
- 2015-09 Randy Klaassen(UT)
HCI Perspectives on Behavior Change Support Systems
- 2015-10 Henry Hermans (OUN)
OpenU: design of an integrated system to support lifelong learning
- 2015-11 Yongming Luo(TUE)
Designing algorithms for big graph datasets: A study of computing bisimulation and joins
- 2015-12 Julie M. Birkholz (VU)
Modi Operandi of Social Network Dynamics:
The Effect of Context on Scientific Collaboration Networks
- 2015-13 Giuseppe Procaccianti(VU)
Energy-Efficient Software
- 2015-14 Bart van Straalen (UT)
A cognitive approach to modeling bad news conversations
- 2015-15 Klaas Andries de Graaf (VU)
Ontology-based Software Architecture Documentation

Bibliography

- [1] ISO/IEC/IEEE systems and software engineering – vocabulary. *ISO/IEC/IEEE 24765:2010(E)*, pages 1–418, 2010. (Cited on page 1.)
- [2] ISO/IEC/IEEE systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E)*, pages 1–46, 2011. (Cited on pages 18, 40, 117, 124, 127, and 139.)
- [3] Steve Adolph, Wendy Hall, and Philippe Kruchten. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, 16 (4)(4):487–513, 2011. (Cited on page 61.)
- [4] Art Akerman and Jeff Tyree. Using ontology to support development of software architectures. *IBM Systems Journal*, 45(4):813–825, 2006. (Cited on pages 4 and 46.)
- [5] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. MIT Press, second edition, 2008. (Cited on page 46.)
- [6] Soren Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki a tool for social, semantic collaboration. In *5th International Semantic Web Conference ISWC2006*, pages 736–749. Springer LNCS, 2006. (Cited on pages 48 and 148.)
- [7] Muhammad Ali Babar and Ian Gorton. A tool for managing software architecture knowledge. In *Workshop on SHARing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (SHARK-ADI)*, pages 11–18. IEEE, 2007. (Cited on page 54.)
- [8] Lenin Babu T., M. Seetha Ramaiah, T. V. Prabhakar, and D. Rambabu. Archvoc-towards an ontology for software architecture. In *Workshop on SHARing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (SHARK-ADI)*, pages 5–11. IEEE, 2007. (Cited on pages 46 and 118.)
- [9] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, third edition, 2012. (Cited on pages 2, 3, 40, 42, 50, 58, 71, 124, 139, and 141.)
- [10] Len Bass, Rick Kazman, and Ipek Ozkaya. Developing architectural documentation for the hadoop distributed file system. In *Open Source Systems: Grounding Research*, volume 365 of *IFIP Advances in Information and Communication Technology*, pages 50–61. Springer, 2011. (Cited on page 111.)

BIBLIOGRAPHY

- [11] Grady Booch. Architecture reviews. *IEEE Software*, 27(3):96, 95, 2010. (Cited on page 122.)
- [12] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571 – 583, 2007. (Cited on page 8.)
- [13] Andrew Brooks, Marc Roper, Murray Wood, John W. Daly, and James Miller. Replication’s role in software engineering. In *Guide to Advanced Empirical Software Engineering*, pages 365–379. Springer, 2008. (Cited on page 115.)
- [14] Michel Buffa, Fabien Gandon, Guillaume Ereteo, Peter Sander, and Catherine Faron. Sweetwiki: A semantic wiki. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):84 – 97, 2008. (Cited on page 43.)
- [15] Janet E. Burge and David C. Brown. Software engineering using rationale. *Journal of Systems and Software*, 81(3):395–413, 2008. (Cited on page 54.)
- [16] Tobias Bürger, Elena Simperl, Stephan Wölger, and Simon Hangl. Using cost-benefit information in ontology engineering projects. In *Context and Semantics for Knowledge Management*, pages 61–90. Springer, 2011. (Cited on page 69.)
- [17] Gul Calikli, Ayse Bener, and Berna Arslan. An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers. In *International Conference on Software Engineering (ICSE)*, pages 187–190. IEEE, 2010. (Cited on page 36.)
- [18] Rafael Capilla, Francisco Nava, Sandra Pérez, and Juan C. Dueñas. A web-based tool for managing architectural design decisions. *ACM SIGSOFT Software Engineering Notes*, 31(5), 2006. (Cited on page 54.)
- [19] Hsinchun Chen and Vasant Dhar. Cognitive process as a basis for intelligent retrieval systems design. *Information Processing & Management*, 27(5):405 – 432, 1991. (Cited on pages 8, 21, 34, and 35.)
- [20] Jie-Cherng Chen and Sun-Jen Huang. An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software*, 82(6):981–992, 2009. (Cited on page 3.)
- [21] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, and Judith Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, second edition, 2010. (Cited on pages 2, 3, 4, 40, 42, 84, 141, 145, and 146.)

- [22] Jeff Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, 1987. (Cited on page 43.)
- [23] Remco C. de Boer and Hans van Vliet. Architectural knowledge discovery with latent semantic analysis: Constructing a reading guide for software product audits. *Journal of Systems and Software*, 81(9):1456–1469, 2008. (Cited on pages 42 and 147.)
- [24] Remco C. de Boer and Hans van Vliet. Writing and reading software documentation: How the development process may affect understanding. In *ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE)*, pages 40–47. IEEE, 2009. (Cited on page 35.)
- [25] Klaas Andries de Graaf. Annotating software documentation in semantic wikis. In *Workshop on Exploiting semantic annotations in information retrieval (ESAIR)*, pages 5–6. ACM, 2011. (Cited on page 13.)
- [26] Klaas Andries de Graaf, Peng Liang, Antony Tang, Willem Robert van Hage, and Hans van Vliet. An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7):1053 – 1064, 2014. (Cited on pages 13 and 128.)
- [27] Klaas Andries de Graaf, Peng Liang, Antony Tang, and Hans van Vliet. The impact of prior knowledge on searching in software documentation. In *ACM Symposium on Document Engineering (DocEng)*, pages 189–198. ACM, 2014. (Cited on page 12.)
- [28] Klaas Andries de Graaf, Peng Liang, Antony Tang, and Hans van Vliet. Supporting architecture documentation: A comparison of two ontologies for knowledge retrieval. In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2015 - <http://dx.doi.org/10.1145/2745802.2745804>. (Cited on page 13.)
- [29] Klaas Andries de Graaf, Peng Liang, Antony Tang, and Hans van Vliet. How organisation of architecture documentation affects architectural knowledge retrieval. *Science of Computer Programming - Special Issue on Knowledge-based Software Engineering*, March 2016 - under review. (Cited on pages 12 and 13.)
- [30] Klaas Andries de Graaf, Antony Tang, Peng Liang, and Hans van Vliet. Ontology-based software architecture documentation. In *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, pages 121–130. IEEE, 2012. (Cited on pages 13, 72, and 95.)

BIBLIOGRAPHY

- [31] Jorge Andrés Díaz-Pace, Matias Nicoletti, Silvia N. Schiaffino, Christian Villavicencio, and Luis Emiliano Sanchez. A stakeholder-centric optimization strategy for architectural documentation. In *International Conference on Model and Data Engineering (MEDI)*, pages 104–117. Springer LNCS, 2013. (Cited on pages 4, 61, and 84.)
- [32] Andrew Dillon, Cliff McKnight, and John Richardson. Navigation in hypertext: A critical review of the concept. In *International Conference on Human-Computer Interaction (INTERACT)*, pages 587–592. North-Holland Publishing Co., 1990. (Cited on page 43.)
- [33] Wei Ding, Peng Liang, Antony Tang, and Hans van Vliet. Knowledge-based approaches in software documentation: A systematic literature review. *Information and Software Technology*, 56(6):545 – 567, 2014. (Cited on page 35.)
- [34] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer, 2008. (Cited on pages 7 and 8.)
- [35] K. Anders Ericsson and Herbert A. Simon. *Protocol Analysis: Verbal Reports as Data*. MIT Press, revised edition, 1993. (Cited on pages 8 and 21.)
- [36] Davide Falessi, Lionel C. Briand, Giovanni Cantone, Rafael Capilla, and Philippe Kruchten. The value of design rationale information. *ACM Transactions on Software Engineering and Methodology*, 22(3):21:1–21:32, 2013. (Cited on page 138.)
- [37] Andy Field. *Discovering Statistics Using SPSS - 2nd Edition*. Sage Publications, 2005. (Cited on page 116.)
- [38] Golara Garousi, Vahid Garousi-Yusifoglu, Guenther Ruhe, Junji Zhi, Mahmoud Moussavi, and Brian Smith. Usage and usefulness of technical software documentation: An industrial case study. *Information and Software Technology*, 57(0):664 – 682, 2015. (Cited on page 111.)
- [39] T. R. Girill and Clement H. Luk. Hierarchical search support for hypertext on-line documentation. *Man-Machine Studies*, 36(4):571–585, 1992. (Cited on page 43.)
- [40] Barney G. Glaser and Anselm L. Strauss. *The Discovery of Grounded Theory*. Weidenfeld and Nicolson, 1967. (Cited on pages 8 and 61.)
- [41] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199 – 220, 1993. (Cited on page 45.)

- [42] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, December 1995. (Cited on pages 12, 125, and 128.)
- [43] Michael Grüninger and Mark S. Fox. Methodology for the Design and Evaluation of Ontologies. International Joint Conference on Artificial Intelligence (IJCAI), Workshop on Basic Ontological Issues in Knowledge Sharing, 1995. (Cited on page 79.)
- [44] Jin Guo, Jane Cleland-Huang, and Brian Berenbach. Foundations for an expert system in domain-specific traceability. In *International Requirements Engineering Conference (RE)*, pages 42–51, 2013. (Cited on page 147.)
- [45] Hans-Jörg Happel and Stefan Seedorf. Ontobrowse: A semantic wiki for sharing knowledge about software architectures. In *SEKE*, pages 506–512, 2007. (Cited on page 59.)
- [46] Hans-Jörg Happel and Stefan Seedorf. Documenting service-oriented architectures with ontobrowse semantic wiki. In *PRIMIUM*, volume 328, 2008. (Cited on page 54.)
- [47] Martin Hepp. Possible ontologies: How reality constrains the development of relevant ontologies. *IEEE Internet Computing*, 11 (1)(1):90–96, jan-feb 2007. (Cited on pages 58 and 70.)
- [48] Bart Hoenderboom and Peng Liang. A survey of semantic wikis for requirements engineering. Technical report, SEARCH, University of Groningen, 2009. (Cited on page 48.)
- [49] Christine Hofmeister, Robert Nord, and Dilip Soni. *Applied software architecture*. Addison-Wesley, 2000. (Cited on pages 3 and 42.)
- [50] Hilary J. Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research methods in computing: What are they, and how should we teach them? *SIGCSE Bull.*, 38(4):96–114, 2006. (Cited on pages 8 and 9.)
- [51] Anton Jansen, Paris Avgeriou, and Jan Salvador van der Ven. Enriching software architecture documentation. *J. Syst. Softw.*, 82(8):1232–1248, August 2009. (Cited on pages 4, 42, 59, 78, 118, 122, 127, and 141.)
- [52] Anton Jansen and Jan Bosch. Software architecture as a set of architectural design decisions. In *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 109–120. IEEE, 2005. (Cited on page 54.)
- [53] Muhammad Atif Javed and Uwe Zdun. The supportive effect of traceability links in architecture-level software understanding: Two controlled

BIBLIOGRAPHY

- experiments. In *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 215–224. IEEE, 2014. (Cited on page 41.)
- [54] Andreas Jedlitschka, Marcus Ciolkowski, and Dietmar Pfahl. Reporting Experiments in Software Engineering. In *Guide to Advanced Empirical Software Engineering*, pages 201–228. Springer, 2008. (Cited on page 113.)
- [55] Barbara A. Kitchenham and Shari L. Pfleeger. Personal opinion surveys. In *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer, 2008. (Cited on page 8.)
- [56] Andrew J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering*, 32(12):971–987, 2006. (Cited on page 36.)
- [57] Henk Koning and Hans van Vliet. Real-life IT architecture design reports and their relation to IEEE std 1471 stakeholders and concerns. *Automated Software Engineering*, 13(2):201–223, 2006. (Cited on pages 40, 124, 139, and 147.)
- [58] Mikko Korkala and Frank Maurer. Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software*, 95(0):122 – 140, 2014. (Cited on page 36.)
- [59] Konstantinos Kotis and George Vouros. Human-centered ontology engineering: The hcome methodology. *Knowledge and Information Systems*, 10(1)(1):109–131, 2006. (Cited on page 80.)
- [60] Philippe Kruchten. An Ontology of Architectural Design Decisions in Software Intensive Systems. In *2nd Groningen Workshop Software Variability*, pages 54–61, October 2004. (Cited on pages 4 and 46.)
- [61] Philippe Kruchten. Contextualizing agile software development. *Journal of Software: Evolution and Process*, 25(4):351–361, 2013. (Cited on page 67.)
- [62] John A. Kunze and Thomas Baker. Dublin core metadata element set, version 1.1. Technical Report RFC 5013, Internet Engineering Task Force, 2007. (Cited on pages 50 and 51.)
- [63] John K. Kyaruzi and Jan van Katwijk. Beyond components-connections-constraints: Dealing with software architecture difficulties. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 235–242. IEEE, 1999. (Cited on page 46.)

- [64] Patricia Lago and Paris Avgeriou. First workshop on sharing and reusing architectural knowledge. *ACM SIGSOFT Software Engineering Notes*, 31(5):32–36, 2006. (Cited on page 2.)
- [65] Thomas D. LaToza, Gina Venolia, and Robert DeLine. Maintaining mental models: A study of developer work habits. In *International Conference on Software Engineering (ICSE)*, pages 492–501. ACM, 2006. (Cited on pages 15, 16, and 35.)
- [66] Timothy C. Lethbridge, Janice Singer, and Andrew Forward. How software engineers use documentation: The state of the practice. *IEEE Softw.*, 20(6):35–39, November 2003. (Cited on pages 16 and 42.)
- [67] Peng Liang and Paris Avgeriou. Tools and technologies for architecture knowledge management. In *Software Architecture Knowledge Management*, pages 91–111. Springer, 2009. (Cited on page 54.)
- [68] Claudia López, Víctor Codocedo, Hernán Astudillo, and Luiz Marcio Cysneiros. Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach. *Science of Computer Programming*, 77(1):66–80, January 2012. (Cited on pages 4, 46, 118, 122, 141, and 146.)
- [69] Claudia López, Pablo Inostroza, Luiz Marcio Cysneiros, and Hernán Astudillo. Visualization and comparison of architecture rationale with semantic web technologies. *J. Syst. Softw.*, 82(8):1198–1210, August 2009. (Cited on page 45.)
- [70] Jocelyne Nanard, Marc Nanard, Anne-Marie Massotte, Alain Djemaa, Alain Joubert, Henri Betaille, and Jacques Chauché. Integrating knowledge-based hypertext and database for task-oriented access to documents. pages 721–732. Springer LNCS, 1993. (Cited on page 43.)
- [71] Peter Naur and Brian Randell. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968*. NATO Scientific Affairs Division, Brussels, 1969. (Cited on page 1.)
- [72] A.J. Neumann. Management guide for software documentation. In *NBS Special Publication 500-87*. NBS, 1982. (Cited on page 3.)
- [73] Allen Newell and Herber A. Simon. *Human Problem Solving*. Prentice Hall, 1972. (Cited on pages 21 and 34.)

BIBLIOGRAPHY

- [74] Raymond S. Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2(2):175–220, 1998. (Cited on page 32.)
- [75] Antonio De Nicola, Michele Missikoff, and Roberto Navigli. A proposal for a unified process for ontology building: Upon. In *Database and Expert Systems Applications - DEXA*, pages 655–664. Springer LNCS, 2005. (Cited on page 79.)
- [76] Robert Nord, Paul Clements, David Emery, and Rich Hilliard. A structured approach for reviewing architecture documentation. Technical Report CMU/SEI-2009-TN-030, SEI, Software Engineering Institute, Carnegie Mellon University, 2009. (Cited on page 127.)
- [77] David Lorge Parnas. Precise Documentation: The Key to Better Software. In *The Future of Software Engineering*, chapter 8, pages 125–148. Springer, 2011. (Cited on pages 2, 3, 4, 15, 16, 34, 41, and 141.)
- [78] David Lorge Parnas and Paul Clements. A rational design process: How and why to fake it. In *Formal Methods and Software Development*, pages 80–100. Springer LNCS, 1985. (Cited on pages 3 and 40.)
- [79] H. Sofia Pinto, Steffen Staab, Christoph Tempich, and York Sure. Distributed engineering of ontologies (diligent). In *Semantic Web and Peer-to-Peer*, pages 303–322. Springer LNCS, 2006. (Cited on page 79.)
- [80] Dominik Rost, Matthias Naab, Crescencio Lima, and Christina von Flach Garcia Chavez. Software architecture documentation for developers: A survey. In *European Conference on Software Architecture (ECSA)*, pages 72–88. Springer LNCS, 2013. (Cited on pages 3, 4, 15, 41, 42, 84, 123, and 141.)
- [81] Nick Rozanski and Eóin Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2005. (Cited on page 40.)
- [82] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2)(2):131–164, 2009. (Cited on pages 8 and 72.)
- [83] Widura Schwittek and Stefan Eicker. Communicating architectural knowledge: Requirements for software architecture knowledge management tools. In *European Conference on Software Architecture (ECSA)*, pages 457–463. Springer LNCS, 2010. (Cited on pages 4 and 41.)

- [84] Mojtaba Shahin, Peng Liang, and Mohammad Reza Khayyambashi. Architectural design decision: Existing models and tools. In *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 293–296. IEEE, 2009. (Cited on page 46.)
- [85] Mojtaba Shahin, Peng Liang, and Zengyang Li. Architectural design decision visualization for architecture design: preliminary results of a controlled experiment. In *Proceedings of the 5th European Conference on Software Architecture (ECSA): Companion Volume*, pages 2:1–2:8. ACM, 2011. (Cited on page 41.)
- [86] Steven J. Shute and Philip J. Smith. Knowledge-based search tactics. *Information Processing & Management*, 29(1):29 – 45, 1993. (Cited on page 35.)
- [87] Elena Paslaru Bontas Simperl and Christoph Tempich. Ontology engineering: a reality check. In *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences*, pages 836–854. Springer LNCS, 2006. (Cited on page 59.)
- [88] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001. (Cited on page 94.)
- [89] Carlos Solis and Nour Ali. An experience using a spatial hypertext wiki. In *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia (HT)*, pages 133–142. ACM, 2011. (Cited on page 43.)
- [90] Carlos Solis, Nour Ali, and Muhammad Ali Babar. A spatial hypertext wiki for architectural knowledge management. In *Workshop on Wikis for Software Engineering (WIKIS4SE)*, pages 36–46, 2009. (Cited on page 43.)
- [91] Webb Stacy and Jean MacMillan. Cognitive bias in software engineering. *Commun. ACM*, 38(6):57–63, 1995. (Cited on pages 16, 33, and 36.)
- [92] Christoph Johann Stettina and Werner Heijstek. Necessary and neglected?: An empirical study of internal documentation in agile software development teams. In *Proceedings of the 29th ACM International Conference on Design of Communication (SIGDOC)*, pages 159–166. ACM, 2011. (Cited on page 2.)
- [93] Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage, second edition, 1998. (Cited on pages 8, 61, 64, 65, 66, and 73.)
- [94] Moon Ting Su. Capturing exploration to improve software architecture documentation. In *Proceedings of the Fourth European Conference on Software Architecture (ECSA)*, pages 17–21. ACM, 2010. (Cited on page 147.)

BIBLIOGRAPHY

- [95] Moon Ting Su, Christian Hirsch, and John Hosking. Kaitorobase: Visual exploration of software architecture documents. In *International Conference on Automated Software Engineering (ASE)*, pages 657–659. IEEE, 2009. (Cited on pages 4 and 54.)
- [96] Moon Ting Su, Ewan Tempero, John Hosking, and John Grundy. A study of architectural information foraging in software architecture documents. In *Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, pages 141–150. IEEE, 2012. (Cited on page 36.)
- [97] York Sure, Steffen Staab, and Rudi Studer. On-to-knowledge methodology (otkm). In *Handbook on Ontologies*, pages 117–132. Springer, 2004. (Cited on page 79.)
- [98] York Sure, Christoph Tempich, and Denny Vrandeic. Ontology engineering methodologies. In *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, pages 171–190. Wiley, UK, 2006. (Cited on page 70.)
- [99] Damien Andrew Tamburri. An architecture description viewpoint wiki based on the semantic web paradigm. Master’s thesis, Department of Computer Science, VU University Amsterdam, 2010. (Cited on pages 48, 117, and 147.)
- [100] Antony Tang. Software designers, are you biased? In *Proceedings of the 6th International Workshop on SHaring and Reusing Architectural Knowledge, (SHARK)*, pages 1–8. ACM, 2011. (Cited on page 35.)
- [101] Antony Tang, Muhammad Ali Babar, Ian Gorton, and Jun Han. A survey of architecture design rationale. *Journal of Systems and Software*, 79(12):1792–1804, 2006. (Cited on pages 2 and 3.)
- [102] Antony Tang, Yan Jin, and Jun Han. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, 80(6):918–934, 2007. (Cited on page 54.)
- [103] Antony Tang, Peng Liang, Viktor Clerc, and Hans van Vliet. Supporting co-evolving architectural requirements and design through traceability and reasoning. In *Relating Software Requirements to Software Architecture*, pages 59 – 85. Springer, 2011. (Cited on pages 46, 122, and 145.)
- [104] Antony Tang, Peng Liang, and Hans van Vliet. Software architecture documentation: The road ahead. In *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 252–255. IEEE, 2011. (Cited on pages 46, 51, 75, 89, and 90.)

- [105] Peri Tarr, Harold Ossher, William Harrison, and Stanley M. Sutton, Jr. N degrees of separation: multi-dimensional separation of concerns. In *International Conference on Software Engineering (ICSE)*, pages 107–119. ACM, 1999. (Cited on page 39.)
- [106] Manfred Thüring, Jörg M. Haake, and Jörg Hannemann. What’s eliza doing in the chinese room? incoherent hyperdocuments and how to avoid them. In *ACM Conference on Hypertext (HYPERTEXT)*, pages 161–177. ACM, 1991. (Cited on page 43.)
- [107] Efraim Turban and Jay E. Aronson. *Decision Support Systems and Intelligent Systems*. Prentice Hall, sixth edition, 2000. (Cited on pages 63 and 70.)
- [108] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974. (Cited on pages 16, 27, and 28.)
- [109] Cathy Urquhart and Walter Fernandez. Grounded theory method: the researcher as blank slate and other myths. *International Conference on Information Systems (ICIS)*, pages 457–464, 2006. (Cited on page 61.)
- [110] Mike Uschold and Martin King. Towards a methodology for building ontologies. *International Joint Conference on Artificial Intelligence (IJCAI), Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995. (Cited on page 79.)
- [111] Jan Salvador van der Ven, Anton Jansen, Paris Avgeriou, and Dieter K. Hammer. Using architectural decisions. In *Conference on the Quality of Software Architectures (QoSA)*. Karlsruhe University Press, 2006. (Cited on page 42.)
- [112] Uwe van Heesch, Paris Avgeriou, and Rich Hilliard. A documentation framework for architecture decisions. *Journal of Systems and Software*, 85(4):795 – 820, 2012. (Cited on page 40.)
- [113] Richard L. Van Horn. Empirical studies of management information systems. *SIGMIS Database*, 5(2-3-4):172–182, 1973. (Cited on page 8.)
- [114] Cornelis Joost van Rijsbergen. *Information Retrieval*. Butterworths & Co, second edition, 1979. (Cited on pages 94 and 130.)
- [115] Maarten W. van Someren, Yvonne F. Barnard, and Jacobijn A.C. Sandberg. *The Think Aloud Method - A practical guide to modelling cognitive processes*. Academic Press London, 1994. (Cited on pages 8 and 20.)

BIBLIOGRAPHY

- [116] Hans van Vliet. *Software engineering - principles and practice*. Wiley, third edition, 2008. (Cited on page 2.)
- [117] Marcello Visconti and Curtis R. Cook. Assessing the state of software documentation practices. In *Product Focused Software Process Improvement*, LNCS, pages 485–496. Springer, 2004. (Cited on page 2.)
- [118] Weigang Wang and Roy Rada. Experiences with semantic net based hypermedia. *International Journal of Human-Computer Studies*, 43(3):419 – 439, 1995. (Cited on page 43.)
- [119] Christopher A. Welty and David A. Ferrucci. A formal ontology for reuse of software architecture documents. In *International Conference on Automated Software Engineering (ASE)*, pages 259–270. IEEE, 1999. (Cited on pages 4 and 46.)
- [120] Robert K. Yin. *Case study research: design and methods*. Sage, fourth edition, 2009. (Cited on page 8.)
- [121] John Zachman. The zachman framework for enterprise architecture. *Zachman International*, 2002. (Cited on page 79.)
- [122] Junji Zhi, Vahid Garousi-Yusifoglu, Bo Sun, Golara Garousi, Shawn Shahnewaz, and Guenther Ruhe. Cost, benefits and quality of software development documentation: A systematic mapping. *Journal of Systems and Software*, 99(0):175 – 198, 2015. (Cited on page 111.)

Abbreviations

AK	– Architectural Knowledge
CF	– Contextual Factor
CRM	– Computer Research Method
e.g.	– <i>exempli gratia</i> (example)
et al.	– <i>et alii</i> (and others)
etc.	– <i>et cetera</i> (and the rest, and so forth)
GT	– Grounded Theory
GUI	– Graphical User Interface
HTML	– HyperText Markup Language
i.e.	– <i>id est</i> (that is; in other words)
OWL	– Web Ontology Language
PBG	– Problem Behaviour Graph
R&D	– Research and Development
RDF	– Resource Description Framework
RQ	– Research Question
SA	– Software Architecture
SAD	– Software Architecture Document
SBD	– Software Behaviour Document
SE	– Software Engineering
Sysref	– System Reference Document
URL	– Uniform Resource Locator
URI	– Uniform Resource Identifier
UML	– Unified Modelling Language
WYSIWYG	– What You See Is What You Get