Science of Computer Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

How organisation of architecture documentation affects architectural knowledge retrieval

K.A. de Graaf [a], P. Liang [b,*], A. Tang [c], H. van Vliet [a]

[a] VU University Amsterdam, Amsterdam, The Netherlands
[b] Wuhan University, Wuhan, China
[c] Swinburne University of Technology, Melbourne, Australia

ARTICLE INFO

Article history:
Received 11 March 2015
Received in revised form 21 September 2015
Accepted 22 October 2015
Available online xxxx

Keywords:
Software architecture documentation
Architectural knowledge retrieval
Software ontologies
Semantic wiki
Ontology-based documentation

ABSTRACT

A common approach to software architecture documentation in industry projects is the use of file-based documents. This approach offers a single-dimensional arrangement of the architectural knowledge. Knowledge retrieval from file-based architecture documentation is efficient if the organisation of knowledge supports the needs of the readers; otherwise it can be difficult. In this paper, we compare the organisation and retrieval of architectural knowledge in a file-based documentation approach and an ontology-based documentation approach. The ontology-based approach offers a multi-dimensional organisation of architectural knowledge 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 architectural knowledge from the different organisations of knowledge. We found that the use of better knowledge organisation correlates with the efficiency and effectiveness of AK retrieval. Professionals who used the knowledge organisation found this beneficial.

1. Introduction

It is recognised by Bass et al. in [1] and Clements et al. in [2] that even a perfect software architecture is essentially useless if it is not understood; proper documentation should have enough detail, no ambiguity, and it must be organised such that users can quickly find information and answer their questions [3]. Documentation of software architecture serves three important purposes: it is used for education, system analysis, and it is the primary vehicle for stakeholder communication [2]. Kruchten suggests that if an architecture is not documented, it does not exist [4].

Architectural Knowledge (AK) is contained in Software Architecture (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*" [5].

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:

* Corresponding author.
E-mail addresses: kadegraaf@gmail.com (K.A. de Graaf), liangp@whu.edu.cn (P. Liang), atang@swin.edu.au (A. Tang), hans@cs.vu.nl (H. van Vliet).

http://dx.doi.org/10.1016/j.scico.2015.10.014
0167-6423/

- 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 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 [2,6], yet documentation in industry is often 'one-size-fits-all' and does not serve specific users and their tasks well [7].

We evaluate an approach that addresses the issues with the linear organisation of AK in file-based documentation. We use a lightweight software ontology and a semantic wiki tool to document software architecture, as proposed in [8]. This ontology-based approach provides multiple ways for users to retrieve AK using the relationships and knowledge defined in the ontology.

We used two controlled industry experiments to compare the ontology-based approach to the file-based approach to investigate how the organisation of AK influences the effectiveness and efficiency of AK retrieval. We measured the time-efficiency and effectiveness (in terms of precision and recall) of software professionals answering questions representative of their daily work. Both approaches were configured for the experiment using available technologies and materials.

We investigated the reasons why software professionals retrieved AK efficiently and effectively. For example, whether certain information in the AK organisations provided clues about the navigation path to relevant types of AK and relationships between AK.

We identified which file-based and ontology-based AK organisation supported software professionals when searching for the AK necessary to answer their questions. The usage of supporting AK organisation was measured from the individual search actions of the software professionals. We then tested whether the use of AK organisation that supports a question has a correlation with the efficiency and effectiveness of answering that question.

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 multi-dimensional ontology-based AK organisation and single-dimensional file-based AK organisation in two industrial experiments.
- Identify how and why AK organisation results in efficient and effective AK retrieval.
- Introduce an ontology-based documentation approach that addresses issues with AK organisation and retrieval in file-based documentation.

In Section 2 we provide the background on SA documentation and its issues and challenges. Section 3 describes our ontology-based approach. Section 4 details on the experiment setup and findings. Section 5 describes how AK retrieval is influenced by AK organisation. Section 6 reports a qualitative evaluation of the documentation approaches and experiment. Section 7 reports a cost–benefit analysis of adopting ontology-based documentation. Threats to validity are discussed in Section 8 and implications of this work are described in Section 9. Related work is discussed in Section 10. Section 11 reports our conclusions and future work.

Replication of the experiment, a questionnaire on experiment results, and an analysis on the use of AK organisation are the major extensions to the work reported in [9].

2. Background

2.1. File-based documentation and its issues

In their highly influential paper on multi-dimensional software decomposition [10], Tarr et al. describe how traditional formalisms in software engineering can only provide a single "dominant" dimension when achieving separation of concerns. Single-dimensional software decomposition 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 [11] 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 [1,2,12].

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

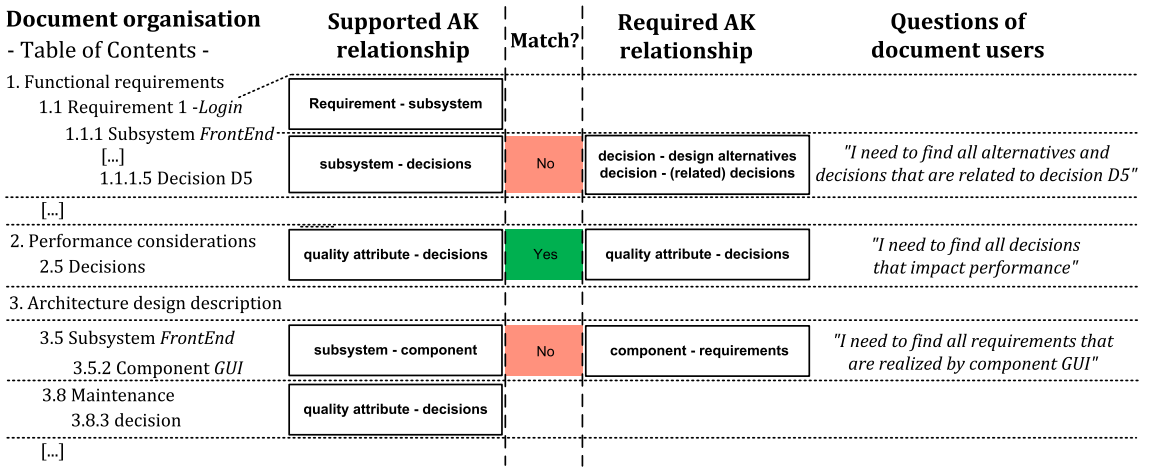

Fig. 1. Mismatch between supported and required AK relationships in file-based document organisation.

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 recognised by Rozanski and Woods in [13], 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 [14]. Moreover, existing approaches for documenting decisions only frame part of the stakeholders concerns related to decisions [15]. 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 Fig. 1. The questions are about certain types of AK, e.g., decisions and requirements, and relationships between AK, e.g., 'impacts' and 'realised 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 realised 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 Fig. 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 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 [3] and organise the AK therein such that it is successfully communicated to users with different backgrounds [16].

Explicitly describing relationships between AK makes the AK traceable. Empirical evidence is given by Shahin et al. in [17] and Javed and Zdun in [18] that improved traceability leads to better architectural understanding. Lack of traceability in SA documentation is considered a major problem in industry practice [7].

In [19] Jansen et al. identify AK retrieval challenges that partially stem from above issues with organising AK. We describe in Appendix A 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 [2]. The original intention of the authors is often lost.
2. *Locating relevant architectural knowledge*
   Knowledge is often spread over multiple documents [20] which makes it hard to locate AK, especially if documents lack finer details.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

3. *Support for traceability between different entities*
   Providing traceability between documentation sources is difficult [21]. 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 [22,1].
6. *Credibility of information*
   AK often changes in large and complex systems and the cost to update is sometimes prohibitive [19]. Documentation is quickly outdated and its users lose confidence in its credibility [23].

Problems related to the above issues and challenges are reported by Rost et al. in a recent survey [7] 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) *i*nconsistent and missing structure, 2) *scattered information*, and 3) *missing traceability*.

2.2. Hypertext documentation and its issues

Hypertext and wiki systems have been used for SA documentation. The use of tags and categories can help to organise knowledge. However tags quickly lose meaning when used arbitrarily. Hypertext is also known as nonlinear text [24], 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 users will be able to understand an organisation of AK by means of hyperlinks.

Several researchers [25,24,26] 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 [27]. Likewise, users of wiki-systems may also experience a lack of structure when navigating and finding relevant information [28].

Hypertext systems address AK retrieval challenges 2 and 3 (in Section 2.1) 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 [29], hyperlinks in knowledge-based hypertext [30], and labelled links in spatial hypertext systems. Solis et al. describe a spatial hypertext systems for AK retrieval in [31] and its qualitative evaluation in [32], which is the only study on using spatial hypertext for AK retrieval that we know of.

3. Ontology-based documentation

The goal of this study is to investigate and compare the use of file-based and ontology-based AK organisation to retrieve documented AK. This section describes our ontology-based approach, how it is used to organise AK, and how it addresses AK retrieval challenges. Our approach, which is evaluated in Sections 4 through 7, consists of the semantic wiki described in [33] and ontologies described in [8,34].

3.1. Software architecture ontologies

"An ontology" explicitly specifies the conceptualisation of a domain [35], i.e. "an ontology" refers to a formal domain model in which concepts and relationships among concepts are described [36]. Ontologies enable a hierarchical classification of interrelated domain concepts and can be represented using a Resource Description Framework (RDF) Schema or the Web Ontology Language (OWL). The use of RDF makes ontologies human readable and machine-interpretable, allowing querying of, and inference over knowledge.

Ontologies, RDF, and OWL are part of the semantic web paradigm which aims to support advanced knowledge management systems in which knowledge can be retrieved via query answering and presented in a human-friendly way [37]. Several ontologies have been recently proposed to express, share, and manage AK. For example, to share architectural design decisions explicitly [38], provide a precise and common vocabulary for making architecture decisions [39,40], and to reuse SA documents [41].

In this study we use the lightweight software ontology from [8] to annotate knowledge in SA documents. The lightweight software ontology is a general-purpose ontology; it contains AK concepts that are commonly documented in a software project [8]. This ontology was built to support use cases around typical activities of architects [42]. The lightweight ontology is designed to be flexible so that it can be adapted for specific application domains. We chose to use the lightweight

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

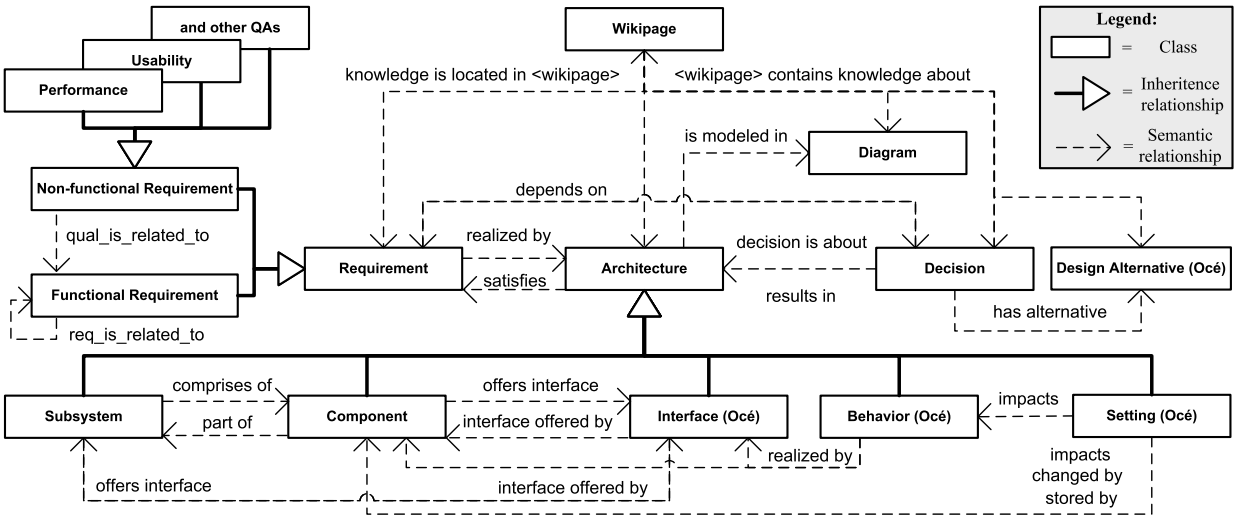

Fig. 2. Software ontology extended for the Océ experiment domain.

software ontology instead of other ontologies because it provides different ways to support AK storage and retrieval, yet is not too costly and time-consuming to enact.

Other general-purpose ontologies for describing commonly documented AK concepts have been proposed in [43,41,44, 39,45]. Many AK concepts in the lightweight software ontology are also described in these other general-purpose ontologies, e.g., requirements in [43], architecture elements such as components, subsystems, and interfaces in [41,44], and all aforementioned AK concepts together with decisions in [39,45].

Fig. 2 depicts the classes and relationships in the lightweight software ontology,[1] together with classes (appended with "(Océ)") that were added to support the AK concepts used in one of the experiment domains (Section 4.3.2 details on this extension). We illustrate the full ontology with a software development scenario (classes are marked boldface and relationships are marked italic):

A software architect makes a **decision** that **non-functional requirement** 'configurability' is *realised 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 *realise* the **behaviour** and details on the **components** or **subsystems** that *offer* the **interfaces**. The **behaviour** can be tested using the **requirements** that it *satisfies* and the **settings** that *impact* it. **Wikipages** *contain knowledge about* the aforementioned AK.

The relationships and classes in the ontology are used for organising AK. Relationships between classes support the 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. We refer to the relationships in the ontology as '*semantic relationships*', because their names and properties convey their meaning.

3.2. ArchiMind semantic Wiki

A semantic wiki allows for navigation of ontology classes and semantic relationships. We used OntoWiki as our semantic wiki tool [46]. OntoWiki is similar to existing wiki systems (e.g., Wikipedia), and additionally offers web-based visualisation and management of (ontology and its instances in) knowledge bases and semantic-enhanced search facilities.

We based our choice for OntoWiki on the evaluation of semantic wikis in [47,48]. In [48] Tamburri used a literature study to identify requirements for a semantic wiki for software knowledge management. OntoWiki satisfied most of these requirements compared to other semantic wikis, most notably requirements for faceted ontology browsing, different views, semantic inference, and social collaboration. Hoenderboom and Liang show in [47] that OntoWiki provides many useful features for requirements engineering, especially semantic search and text annotation.

Version 0.9.5 of OntoWiki was adapted to optimise it for storage and retrieval of SA documentation. The adapted version was named 'ArchiMind' [33]. See http://www.archimind.nl/archimind/ for a demonstration. See Appendix A for a detailed description of ArchiMind and how it addresses the AK retrieval challenges described in Section 2.1.

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

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

Table 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, 1794 paragraphs, and 3183 lines, 13 962 words	46 pages, 20 diagrams, 348 paragraphs, 645 lines, and 8538 words

4. AK retrieval efficiency and effectiveness

4.1. 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 documentation approach and the ontology-based documentation approach.
- **(B)** evaluate the AK retrieval **effectiveness** of the file-based documentation approach and the ontology-based documentation approach.

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. Océ is an international leader in digital document management and a Canon Group company. LaiAn is a software company that provides information system development and integration services for small and medium enterprises and local government. Table 1 details the variations between the experiment domains and the SA documentation used in the experiment.

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.

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.

4.2. Experiment participants

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. At LaiAn we asked technical employees that use the experiment documentation to participate and most agreed to this.

Twenty-six and twenty-two software professionals participated in the experiment at Océ and LaiAn, respectively. These professionals work in various roles, including software engineer, software architect, domain architect, workflow architect, software designer, product- and system test engineer, software project manager, and configuration manager. Table C.3 in Appendix C gives more details on the demographics of the experiment participants at the two companies.

4.3. 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. Fig. 3 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

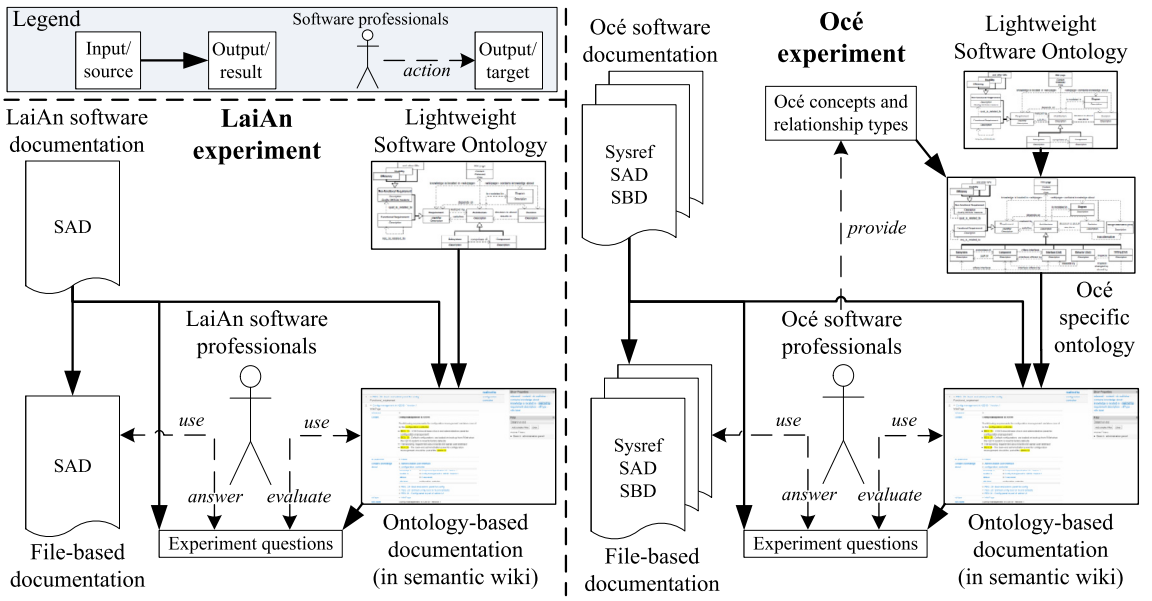

Fig. 3. Construction and use of materials in experiments.

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.

4.3.1. File-based documentation

File-based documents used in the Océ experiment 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 [12]. 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. Three Océ software professionals confirmed that the documents are representative of their usual practice. Question 6 of a questionnaire among experiment participants in Table D.4 also confirms this.

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.

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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

4.3.2. Ontology

The lightweight software ontology from [8] was directly used in the LaiAn experiment. At Océ, we extended the lightweight software ontology to include Océ concepts and relationships types. We used an ontology engineering approach described in [34] 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 Fig. 2) 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. The additional AK concepts (marked boldface) and their relationships (marked italic between parentheses) are; **Behaviour** (*realised by*); **Design Alternative** (*has alternative*); **Setting** (*impacts*, *changed by*, *stored by*); **Interface** (*offers interface*, *interface offered by*).

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. 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 [34].

4.3.3. 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 Fig. 2, namely; 27 wikipages and 3 diagrams; 45 functional and 0 non-functional requirements[2]; 22 decisions; 3 alternative decisions; 19 subsystems; 66 interfaces; 15 components; 8 settings; 6 behaviours.

We identified and annotated 141 AK instances in the LaiAn documents using the lightweight ontology [8], namely; 1 wikipage[3]; 20 diagrams; 65 functional and 2 non-functional requirements (1 system goal was annotated as a requirement); 7 decisions; 21 components; 7 subsystems; 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 realised 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.

4.3.4. 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

[2] Two 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.

[3] Document 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

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.

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 recognisable 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 (realised) by component "XX"?*
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 realised 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

4.4. Experiment hypothesis

We formulate the following alternative hypotheses for experimental goal A and B presented in Section 4.1;

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.

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 ***F*1 score**, introduced by van Rijsbergen in [49], is used for measuring effectiveness:

$$F1\ score = 2 * \frac{Precision\ *\ Recall}{Precision\ +\ Recall}$$

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 [50].

4.5. 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 minimise 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 minimises 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.

Océ participants were asked to think aloud, verbally state their answers, and their satisfaction with answers. LaiAn participants wrote down answers instead of verbalising 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.

4.6. 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 B.2 reports measurements and results[4] of one-tailed tests.

4.6.1. 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 B.2, in column '*p-value* test results', for rows with 'Seconds' in column 'measure'. Consequently, we reject the null hypothesis **H_{0A}** and accept the alternative hypothesis **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 B.2. Consequently, we reject the null hypothesis **H_{0A}** and accept the alternative hypothesis **H_{1A}** for all questions except question 1. The failure to reject the null hypothesis for LaiAn question 1 is explained

[4] Measurements for LaiAn question 1, answered by participant 1, were excluded as we unintentionally asked a slightly different question.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

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.

4.6.2. 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 B.2. Consequently, we reject the null hypothesis $\mathbf{H_{0B}}$ and accept the alternative $\mathbf{H_{1B}}$ 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 B.2. Consequently, we reject the null hypothesis $\mathbf{H_{0B}}$ and accept the alternative $\mathbf{H_{1B}}$ for all LaiAn questions except question 4. $\mathbf{H_{1B}}$ 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 5.1.2 for more details).

5. 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.

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 analysing 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.

5.1. AK organisation

5.1.1. 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-based documentation is organised by ontology classes and by semantic relationships between classes. Figs. 4 and 5 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 Fig. 4). 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 Figs. 4 and 5 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 the paper we use "fitting AK organisation" to refer to the supporting AK organisation that is identified and further investigated.

5.1.2. 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. Figs. 4 and 5 show the average measured efficiency (in seconds) and effectiveness (in F1 score) per question as a means for comparison.

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 documenta-

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

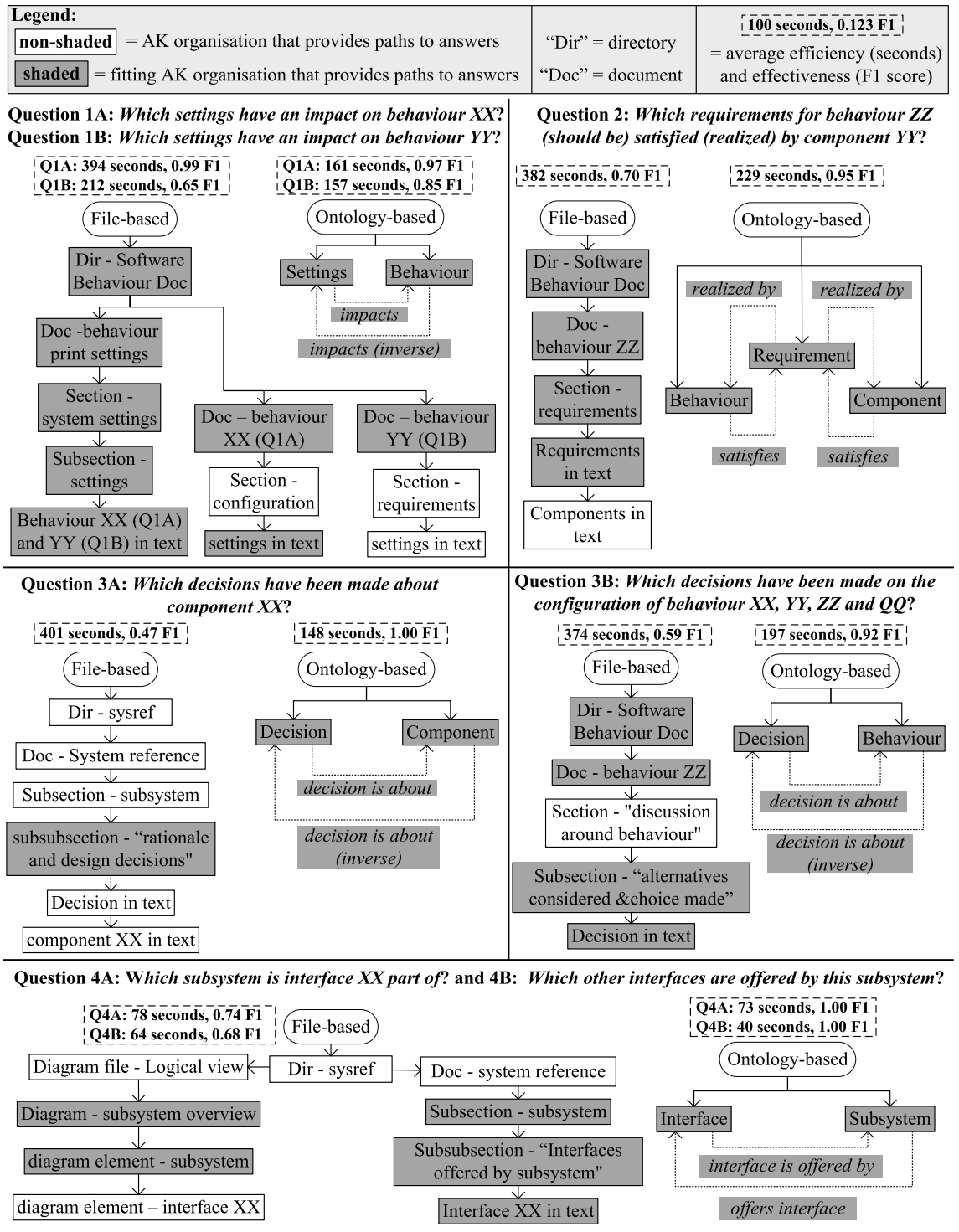

Fig. 4. AK organisation for answering experiment questions at Océ.

tion. This explains why there is no significant difference in effectiveness between the documentation approaches for these questions (also see Section 4.6.2).

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

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

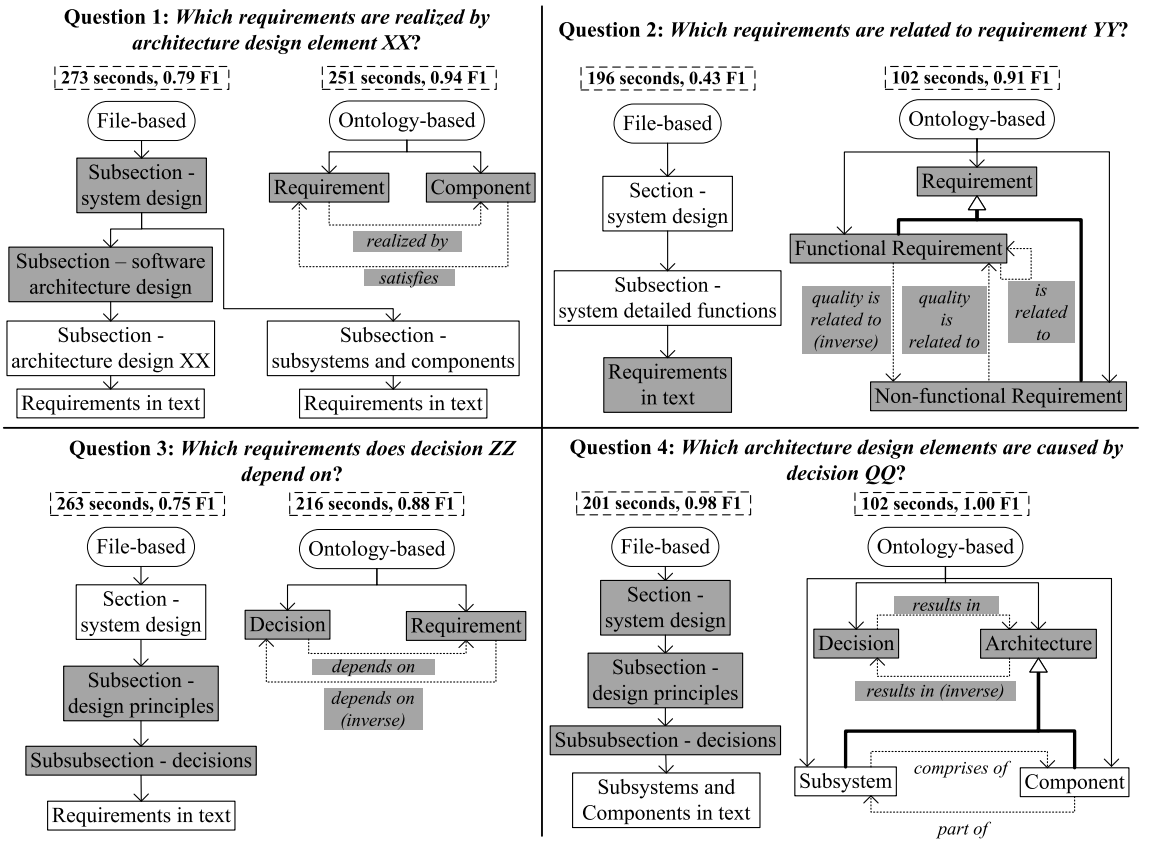

Fig. 5. AK organisation for answering experiment questions at LaiAn.

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*'.

5.2. Use of fitting AK organisation

The analysis in Section 5.1.2 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 analyse 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.

We measured use of AK organisation in about 6000 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[5] to recognise the fitting AK organisation, and 3) the participant follows the fitting AK organisation and navigates to answers.

[5] Based on our observations from the experiment videos we chose to use 3 seconds as the minimum time required for participants to recognise 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

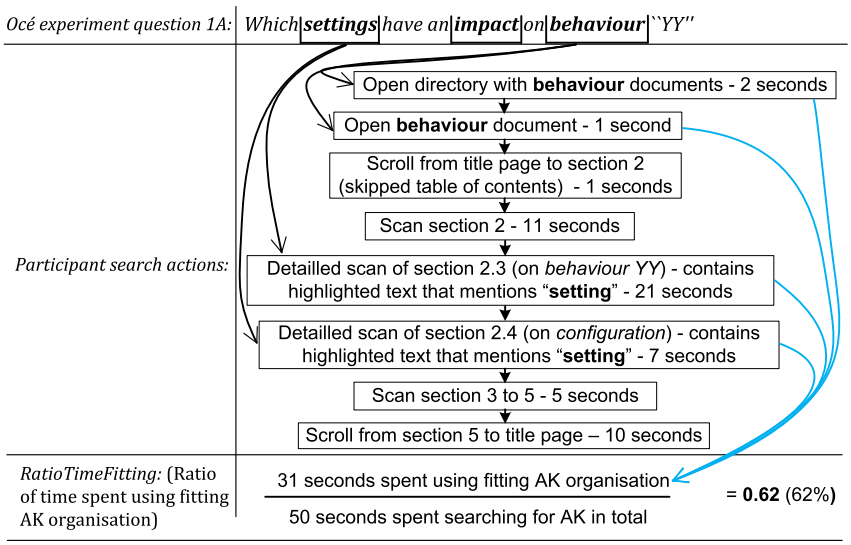

Fig. 6. Example calculation of *RatioTimeFitting* from search actions of a participant using the file-based approach.

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*'.

Fig. 6 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 Fig. 6 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. We use the following hypothesis:

$\mathbf{H_{1C}}$ = *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 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

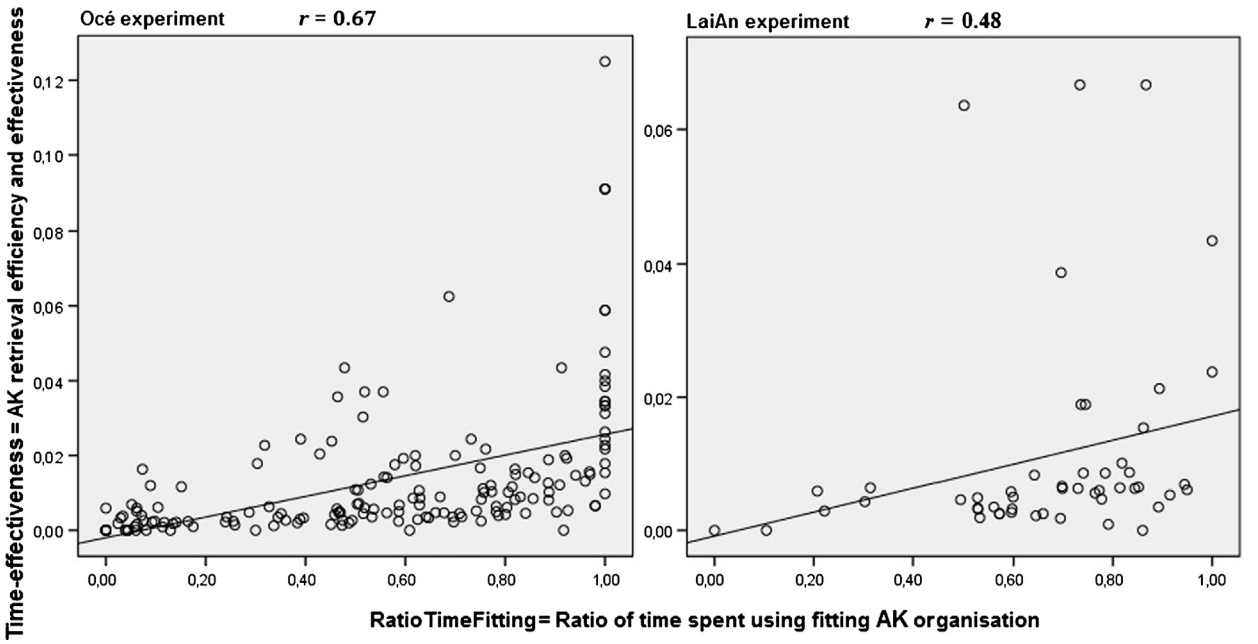

Fig. 7. Correlation between *RatioTimeFitting* and *Time-effectiveness* when answering Océ and LaiAn experiment questions using both documentation approaches.

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 $\mathbf{H_{0C}}$ and accept the alternative hypothesis $\mathbf{H_{1C}}$.

Fig. 7 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. Fig. 7 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).

5.3. 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 recognised the types of AK and relationships between AK.
2. quickly and correctly recognised 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.

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.

The study in [51] reports an in-depth analysis of participants' search behaviour when they used the file-based documentation approach in the Océ experiment. The participants had to deal with search uncertainty when fitting AK organisation was missing in part of the navigation path to the answers they had to find. 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 organisation 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

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

to quickly check the completeness of their answers and remove search uncertainty. This use of ontology-based AK organisation 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) 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 participants 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 the findings about search uncertainty in [51].

6. Qualitative evaluation

After the experiments we asked each participant to fill in a questionnaire, reported in Table D.4, in which the file-based and ontology-based approach (referred to as '*ArchiMind*') are evaluated. Table D.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 D.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.

6.1. 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.

6.2. Evaluation of experiment and ontology

Table D.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.

7. 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 D.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 [52]. A recent systematic literature mapping in [53] 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 [54], 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

7.1. 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 (see Section 4.3.3 for details) 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 D.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 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.

7.2. Cost and benefit in LaiAn project

We spent 4 hours installing and configuring ArchiMind at LaiAn. Around 24 hours were spent to semantically annotate (see Section 4.3.3 for details) 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 D.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 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.

7.3. 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 7.1 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.

8. Threats to validity

In our experiment plan, we accounted for possible threats during the experiment design. We followed guidelines from [55] for reporting the experiments and its threats to validity.

8.1. 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 4.3.4). 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

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.

8.2. 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 D.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).

8.3. External validity

Even though exact replication between experiments in software engineering is unattainable [56], 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 1, indicate that the results are generalisable 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 generalisation, 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 generalisation 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 (MagicDraw) 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 generalisation. 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 SA documentation and in turn increases the potential benefits ontology-based SA documentation.

9. Implications

9.1. Implications for practitioners

We found that the use of AK organisation 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 recognise 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 interrelated 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 organi-

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

sation 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 5 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.

9.2. 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 identified and 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 [12]. 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 [48], seems promising in this light.

10. Related work

Several tools and approaches for managing AK exist such as ADDSS [57], Archium [58], AREL [59], PAKME [60], and SEURAT [61]. These tools and approaches can be used to store, analyse, and retrieve formalised AK with semantics and they support many architecting activities. They differ from our approach in that they are not ontology-based (except for SEURAT [61]) and have less support for storing, managing, and retrieving knowledge contents stored in small and searchable chunks.

Su et al. proposed KaitoroBase [62], a tool for exploring architecture documents, built on freebase semantic wiki. KaitoroBase allows for visualisation 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 realised by a component).

Happel and Seedorf [63] 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 industry validation is a differentiation to our work.

López et al. [45] 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 [43], 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 [45] 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 [45], 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. [19] 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

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

and their relationships. Different tools are used for semantically annotating, viewing, and navigating AK, whereas ArchiMind provides a single integrated interface for this. In [19] 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.

11. Conclusions and future work

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 recognise 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.

Though the results of this study have indicated that ontology-based SA documentation holds a promising future, there are yet many more challenges to overcome. The effort required for semantic annotation may limit adoption of the ontology-based approach. We plan to investigate into semi-automatic input (annotation) of AK using natural language processing, parsing, form-based user input, or rule-based logic (e.g., as proposed in [64]).

Moreover, the level of fitness of an ontology to its intended use can influence its benefits. In a future study we will compare AK retrieval using an ontology built without knowing the questions asked and AK retrieval using an ontology built based on the questions asked. Comparison between ontology-based and hypertext-based approaches to SA documentation will also be future work.

Acknowledgements

The authors wish to thank René Laan, Wim Couwenberg, Pieter Verduin, Amar Kalloe, and the other good folks at Océ R&D for their support, interest to participate in this research, and excellent insights. Also thanks to Jonathan Rebel, Ruben Hartog, and Berend van Veenendaal for adapting OntoWiki. This research has been partially sponsored by the Dutch "Regeling Kenniswerkers (KWR)", project KWR09164, "Stephenson: Architecture knowledge sharing practices in software product lines for print systems" and by the Natural Science Foundation of China (NSFC) project No. 61170025 "KeSRAD: Knowledge-enabled Software Requirements to Architecture Documentation".

Appendix A. ArchiMind semantic Wiki

This section gives a detailed description of ArchiMind and how it addresses the AK retrieval challenges described in Section 2.1.

Fig. A.8 depicts part of the ArchiMind GUI in which red labels highlight different GUI elements. Label *A* highlights the class navigation panel. The class navigation shows the ontology classes in Fig. 2. The subclasses of Architecture and Requirement (denoted by their inheritance relationships in Fig. 2) can be expanded by clicking on the arrowheads. Label *C* highlights a list with instances of class Requirement that were retrieved using the class navigation panel.

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

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 '*realised by*' and '*depends on*'. Faceting allows users to view AK with a certain relationship to the listed AK.

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 an ontology class and its instances are used to store documentation content.

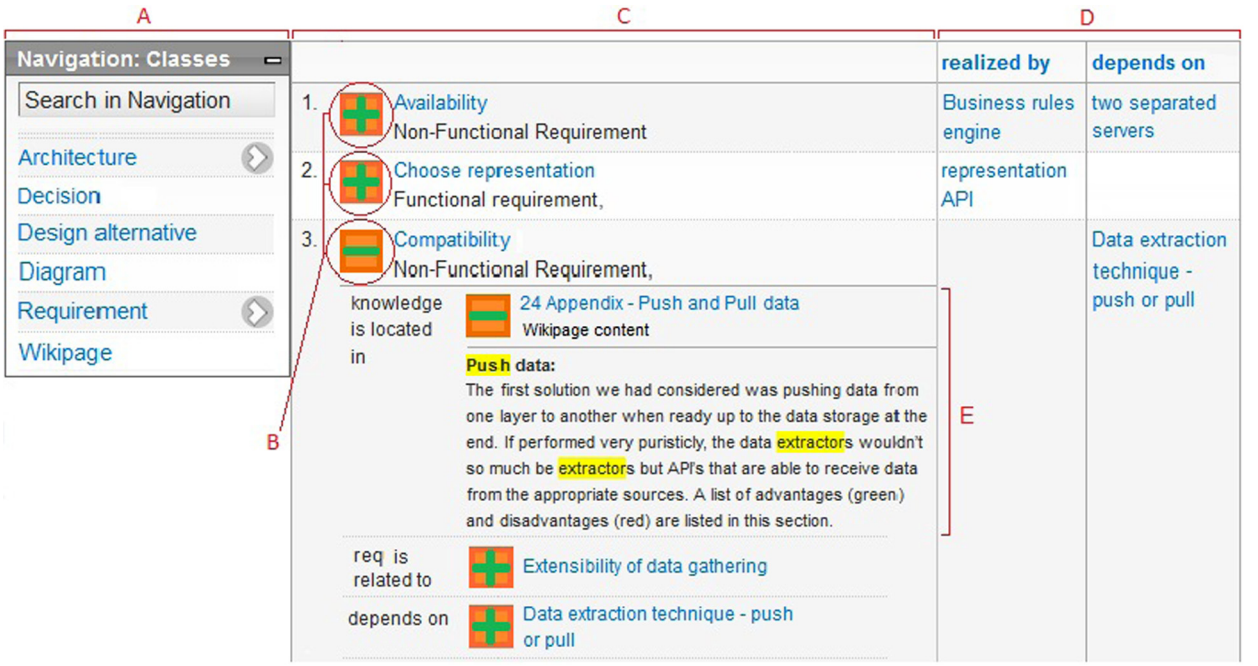

Fig. A.8. AK exploration and faceting in ArchiMind semantic wiki. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

ArchiMind allows for semantic annotation of phrases in documentation content that refer to AK instances, e.g., phrase '*extractor*' refers to an instance of AK type component (see Label *E*). The annotated text in the wikipages is highlighted yellow and, when clicked, a pop-up menu shows the full description of the AK instance, its semantic relationships to other AK instances, and to other wikipages that describe it. See [33] for more details.

The semantic annotations prevent issues with ambiguity, synonyms, homonyms, spelling errors, abbreviations, and context-dependent interpretation of AK in documentation content. This alleviates AK retrieval challenge (1) *Architecture documentation understanding*, described in Section 2.1.

When a phrase is annotated in a wikipage, a semantic relationship is created from the wikipage to the AK instance(s) that the phrase refers to, and vice versa. AK instances become traceable to the documentation content in wikipages that specifies this AK, and vice versa. For example, a user that expands requirement '*Compatibility*' shown in Fig. A.8 (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 also click on annotated phrase '*extractor*' on this wikipage to view a description of this AK instance (component *extractor*). This helps users to locate (sources of) AK descriptions and thereby alleviates AK retrieval challenge (2) *Locating relevant architectural knowledge*.

Semantic relationships in the ontology allow users to see how AK instances are interrelated, e.g., "a requirement is realised by components", and thereby alleviates challenge (3) *Traceability*. If changes are made to an AK instance, e.g., a decision is modified, a user can see what other AK is impacted by the change, e.g., requirements depending on the decision. This alleviates challenge (4) *Change impact analysis*.

Checking for 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 *realised by* architecture elements and the buildability [1] of an architecture can be assessed by following the semantic relationships that indicate dependencies between components.

Dublin Core [65] is used to store documentation meta-data, e.g., date, author, and version of documents. OntoWiki offers version control of knowledge base instances and 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. These features alleviate 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 maintenance effort in a documentation approach is important for its adoption. The use of document meta-data and versioning, 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. Semantic annotation introduces additional costs during maintenance, however, these costs can be minimised using an automatic annotation mechanism.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

Appendix B. Detailed experiment test results

Table B.2
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	*p-Value* test results	Effect size *r*
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	*p-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

Appendix C. Experiment participants details

Table C.3 details on the demographics of the experiment participants at the two companies. 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 also take on the role of requirements engineer and designer; and a project manager may also take on the role of delivery and quality manager.

Table C.3
Demographics of experiment participants at Océ and LaiAn.

Company	Number of participants	Primary role of participants	Average years in role at Océ	Average years in role	Average years working at Océ
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

Company	Number of participants	Primary role of participants	Average years in role at LaiAn	Average years in role	Average years working at LaiAn
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

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

Appendix D. Detailed questionnaire results

Table D.4
Questionnaire about file-based and ontology-based approach.

1: ***When searching for software knowledge, would you evaluate ArchiMind, compared to normal documentation, as:***
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 (centralised) 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-wikipage annotation in the LaiAn experiment (also see Section 4.3.3).
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, authorisation 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.
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.[a]
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.

[a] Océ 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.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

Table D.5
Experiment and ontology evaluation at Océ and LaiAn.

1	***Do you believe in the experiment results? (do the results represent reality or are they artificial)***
Océ: Yes – 4 To a certain extent – 1 No – 0 **LaiAn**: Yes – 5 To a certain extent – 1 No – 0	
2	***Are the experiment results limited to the specific question set used in the experiment?***
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	***Are the experiment questions relevant to your job and representative of the questions you ask during your job?***
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 realised by architecture design) which one Océ and one LaiAn respondent evaluate as irrelevant and not representative.	
4	***Is the ontology model used in the experiment a correct representation of reality?***
Océ: Yes – 3 To a certain extent – 2 No – 0 **LaiAn**: Yes – 4 To a certain extent – 2 No – 0	
5	***Should there be more or less concepts in the model?***
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	***Is it practical to work with the predefined model of software (architecture) knowledge?***
Océ : Yes – 5 To a certain extent – 0 No – 0 **LaiAn**: Yes – 5 To a certain extent – 1 No – 0	
7	***Does the model help in reasoning about what knowledge is in the documents and what should be in documents?***
Océ : Yes – 5 To a certain extent – 0 No – 0 **LaiAn**: Yes – 6 To a certain extent – 0 No – 0	

References

[1] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, 3rd ed., Addison-Wesley, 2012.
[2] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford, Documenting Software Architectures: Views and Beyond, 2nd ed., Addison-Wesley, 2010.
[3] D.L. Parnas, Precise documentation: the key to better software, in: The Future of Software Engineering, Springer, 2011, pp. 125–148.
[4] P. Kruchten, Documentation of software architecture from a knowledge management perspective – design representation, in: Software Architecture Knowledge Management, Springer, 2009, pp. 39–57.
[5] P. Lago, P. Avgeriou, First workshop on sharing and reusing architectural knowledge, Softw. Eng. Notes 31 (2006) 32–36.
[6] J.A. Díaz-Pace, M. Nicoletti, S.N. Schiaffino, C. Villavicencio, L.E. Sanchez, A stakeholder-centric optimization strategy for architectural documentation, in: International Conference on Model and Data Engineering, MEDI, in: LNCS, Springer, 2013, pp. 104–117.
[7] D. Rost, M. Naab, C. Lima, C. von Flach Garcia Chavez, Software architecture documentation for developers: a survey, in: European Conference on Software Architecture, ECSA, in: LNCS, Springer, 2013, pp. 72–88.
[8] A. Tang, P. Liang, H. van Vliet, Software architecture documentation: the road ahead, in: Working IEEE/IFIP Conference on Software Architecture, WICSA, IEEE, 2011, pp. 252–255.
[9] K.A. de Graaf, A. Tang, P. Liang, H. van Vliet, Ontology-based software architecture documentation, in: Joint Working IEEE/IFIP Conference on Software Architecture, WICSA, and European Conference on Software Architecture, ECSA, IEEE, 2012, pp. 121–130.
[10] P. Tarr, H. Ossher, W. Harrison, S.M. Sutton Jr., N degrees of separation: multi-dimensional separation of concerns, in: International Conference on Software Engineering, ICSE, ACM, 1999, pp. 107–119.
[11] D. Parnas, P. Clements, A rational design process: how and why to fake it, in: Formal Methods and Software Development, in: LNCS, vol. 186, Springer, 1985, pp. 80–100.
[12] ISO/IEC/IEEE systems and software engineering – architecture description, ISO/IEC/IEEE 42010:2011(E) (2011) 1–46.
[13] N. Rozanski, E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Addison-Wesley Professional, 2005.
[14] H. Koning, H. van Vliet, Real-life IT architecture design reports and their relation to IEEE std 1471 stakeholders and concerns, Autom. Softw. Eng. 13 (2006) 201–223.
[15] U. van Heesch, P. Avgeriou, R. Hilliard, A documentation framework for architecture decisions, J. Syst. Softw. 85 (2012) 795–820.
[16] W. Schwittek, S. Eicker, Communicating architectural knowledge: requirements for software architecture knowledge management tools, in: European Conference on Software Architecture, ECSA, in: LNCS, vol. 6285, Springer, 2010, pp. 457–463.
[17] M. Shahin, P. Liang, Z. Li, Architectural design decision visualization for architecture design: preliminary results of a controlled experiment, in: Proceedings of the 5th European Conference on Software Architecture, Companion volume, ECSA, ACM, 2011, pp. 2:1–2:8.
[18] M.A. Javed, U. Zdun, The supportive effect of traceability links in architecture-level software understanding: two controlled experiments, in: Working IEEE/IFIP Conference on Software Architecture, WICSA, IEEE, 2014, pp. 215–224.
[19] A. Jansen, P. Avgeriou, J.S. van der Ven, Enriching software architecture documentation, J. Syst. Softw. 82 (2009) 1232–1248.
[20] R.C. de Boer, H. van Vliet, Architectural knowledge discovery with latent semantic analysis: constructing a reading guide for software product audits, J. Syst. Softw. 81 (2008) 1456–1469.
[21] C. Hofmeister, R. Nord, D. Soni, Applied Software Architecture, Addison-Wesley, 2000.
[22] J.S. van der Ven, A. Jansen, P. Avgeriou, D.K. Hammer, Using architectural decisions, in: International Conference on the Quality of Software Architectures, QoSA, Karlsruhe University Press, 2006.
[23] T.C. Lethbridge, J. Singer, A. Forward, How software engineers use documentation: the state of the practice, IEEE Softw. 20 (2003) 35–39.
[24] J. Conklin, Hypertext: an introduction and survey, Computer 20 (1987) 17–41.
[25] A. Dillon, C. McKnight, J. Richardson, Navigation in hypertext: a critical review of the concept, in: International Conference on Human–Computer Interaction, INTERACT, North-Holland Publishing Co., 1990, pp. 587–592.
[26] T.R. Girill, C.H. Luk, Hierarchical search support for hypertext on-line documentation, Int. J. Man-Mach. Stud. 36 (1992) 571–585.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014

[27] M. Thüring, J.M. Haake, J. Hannemann, What's Eliza doing in the Chinese room? Incoherent hyperdocuments and how to avoid them, in: ACM Conference on Hypertext, HYPERTEXT, ACM, 1991, pp. 161–177.
[28] M. Buffa, F. Gandon, G. Ereteo, P. Sander, C. Faron, Sweetwiki: a semantic wiki, Web Semant. Sci. Serv. Agents World Wide Web 6 (2008) 84–97.
[29] W. Wang, R. Rada, Experiences with semantic net based hypermedia, Int. J. Hum.-Comput. Stud. 43 (1995) 419–439.
[30] J. Nanard, M. Nanard, A.-M. Massotte, A. Djemaa, A. Joubert, H. Betaille, J. Chauché, Integrating Knowledge-Based Hypertext and Database for Task-Oriented Access to Documents, LNCS, vol. 720, Springer, 1993, pp. 721–732.
[31] C. Solis, N. Ali, M. Babar, A spatial hypertext wiki for architectural knowledge management, in: Workshop on Wikis for Software Engineering, WIKIS4SE, 2009, pp. 36–46.
[32] C. Solis, N. Ali, An experience using a spatial hypertext wiki, in: Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, HT, ACM, 2011, pp. 133–142.
[33] K.A. de Graaf, Annotating software documentation in semantic wikis, in: Workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR, ACM, 2011, pp. 5–6.
[34] K.A. de Graaf, P. Liang, A. Tang, W.R. van Hage, H. van Vliet, An exploratory study on ontology engineering for software architecture documentation, Comput. Ind. 65 (2014) 1053–1064.
[35] T.R. Gruber, A translation approach to portable ontology specifications, Knowl. Acquis. 5 (1993) 199–220.
[36] C. López, P. Inostroza, L.M. Cysneiros, H. Astudillo, Visualization and comparison of architecture rationale with semantic web technologies, J. Syst. Softw. 82 (2009) 1198–1210.
[37] G. Antoniou, F. van Harmelen, A Semantic Web Primer, second ed., MIT Press, 2008.
[38] M. Shahin, P. Liang, M. Khayyambashi, Architectural design decision: existing models and tools, in: Working IEEE/IFIP Conference on Software Architecture, WICSA, IEEE, 2009, pp. 293–296.
[39] A. Akerman, J. Tyree, Using ontology to support development of software architectures, IBM Syst. J. 45 (2006) 813–825.
[40] P. Kruchten, An ontology of architectural design decisions in software intensive systems, in: 2nd Groningen Workshop Software Variability Management, SVM, 2004, pp. 54–61.
[41] C.A. Welty, D.A. Ferrucci, A formal ontology for re-use of software architecture documents, in: International Conference on Automated Software Engineering, ASE, IEEE, 1999, pp. 259–270.
[42] A. Tang, P. Liang, V. Clerc, H. van Vliet, Traceability in the co-evolution of architectural requirements and design, in: Relating Software Requirements and Architectures, Springer, 2011, pp. 35–60.
[43] T. Lenin Babu, M. Seetha Ramaiah, T.V. Prabhakar, D. Rambabu, Archvoc – towards an ontology for software architecture, in: Workshop on SHAring and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent, SHARK-ADI, IEEE, 2007, pp. 5–11.
[44] J.K. Kyaruzi, J. van Katwijk, Beyond components–connections–constraints: dealing with software architecture difficulties, in: IEEE/ACM International Conference on Automated Software Engineering, ASE, IEEE, 1999, pp. 235–242.
[45] C. López, V. Codocedo, H. Astudillo, L.M. Cysneiros, Bridging the gap between software architecture rationale formalisms and actual architecture documents: an ontology-driven approach, Sci. Comput. Program. 77 (2012) 66–80.
[46] S. Auer, S. Dietzold, T. Riechert, Ontowiki a tool for social, semantic collaboration, in: International Semantic Web Conference, ISWC, in: LNCS, vol. 4273, Springer, 2006, pp. 736–749.
[47] B. Hoenderboom, P. Liang, A survey of semantic wikis for requirements engineering, Technical report, SEARCH, University of Groningen, 2009.
[48] D.A. Tamburri, An architecture description viewpoint wiki based on the semantic web paradigm, Master's thesis, Department of Computer Science, VU University Amsterdam, 2010.
[49] C. van Rijsbergen, Information Retrieval, second ed., Butterworths & Co, 1979.
[50] A. Singhal, Modern information retrieval: a brief overview, IEEE Data Eng. Bull. 24 (2001) 35–43.
[51] K.A. de Graaf, P. Liang, A. Tang, H. van Vliet, The impact of prior knowledge on searching in software documentation, in: ACM Symposium on Document Engineering, DocEng, ACM, 2014, pp. 189–198.
[52] L. Bass, R. Kazman, I. Ozkaya, Developing architectural documentation for the Hadoop distributed file system, in: Open Source Systems: Grounding Research, in: IFIP Advances in Information and Communication Technology, vol. 365, Springer, 2011, pp. 50–61.
[53] J. Zhi, V. Garousi-Yusifoglu, B. Sun, G. Garousi, S. Shahnewaz, G. Ruhe, Cost, benefits and quality of software development documentation: a systematic mapping, J. Syst. Softw. 99 (2015) 175–198.
[54] G. Garousi, V. Garousi-Yusifoglu, G. Ruhe, J. Zhi, M. Moussavi, B. Smith, Usage and usefulness of technical software documentation: an industrial case study, Inf. Softw. Technol. 57 (2015) 664–682.
[55] A. Jedlitschka, M. Ciolkowski, D. Pfahl, Reporting experiments in software engineering, in: Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 201–228.
[56] A. Brooks, M. Roper, M. Wood, J. Daly, J. Miller, Replication's role in software engineering, in: Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 365–379.
[57] R. Capilla, F. Nava, S. Pérez, J.C. Dueñas, A web-based tool for managing architectural design decisions, Softw. Eng. Notes 31 (2006).
[58] A. Jansen, J. Bosch, Software architecture as a set of architectural design decisions, in: Working IEEE/IFIP Conference on Software Architecture, WICSA, IEEE, 2005, pp. 109–120.
[59] A. Tang, Y. Jin, J. Han, A rationale-based architecture model for design traceability and reasoning, J. Syst. Softw. 80 (2007) 918–934.
[60] M.A. Babar, I. Gorton, A tool for managing software architecture knowledge, in: Workshop on SHAring and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent, SHARK-ADI, IEEE, 2007, pp. 11–18.
[61] J.E. Burge, D.C. Brown, Software engineering using rationale, J. Syst. Softw. 81 (2008) 395–413.
[62] M.T. Su, C. Hirsch, J. Hosking, Kaitorobase: visual exploration of software architecture documents, in: International Conference on Automated Software Engineering, ASE, IEEE, 2009, pp. 657–659.
[63] H.-J. Happel, S. Seedorf, Documenting service-oriented architectures with ontobrowse semantic wiki, in: PRIMIUM, vol. 328, 2008.
[64] J. Guo, J. Cleland-Huang, B. Berenbach, Foundations for an expert system in domain-specific traceability, in: International Requirements Engineering Conference, RE, 2013, pp. 42–51.
[65] J. Kunze, T. Baker, Dublin core metadata element set, version 1.1, Technical report RFC 5013, Internet Engineering Task Force, 2007.

Please cite this article in press as: K.A. de Graaf et al., How organisation of architecture documentation affects architectural knowledge retrieval, Sci. Comput. Program. (2015), http://dx.doi.org/10.1016/j.scico.2015.10.014