Ontology-based Software Architecture Documentation

Klaas Andries de Graaf
VU University
ka.de.graaf@vu.nl

Antony Tang
Swinburne University of Technology
atang@ict.swin.edu.au

Peng Liang
Wuhan University
liangp@sklse.org

Hans van Vliet
VU University
hans@cs.vu.nl

***Abstract*—A common approach to software architecture documentation in industry projects is the use of file-based documents. This approach offers a single-dimensional perspective on the architectural knowledge contained. Knowledge retrieval from file-based architecture documentation is efficient if the perspective chosen fits the needs of the readers; it is less so if the perspective does not match the needs of the readers. In this paper we describe an approach aimed at addressing architecture documentation retrieval issues. We have employed a software ontology in a semantic wiki optimized for architecture documentation. We have evaluated this ontology-based approach in a controlled industry experiment involving software professionals. The efficiency and effectiveness of the proposed approach is found to be better than that of the file-based approach.**

***Keywords*-software architecture documentation, software architecture knowledge, architectural knowledge retrieval, software ontologies, semantic wiki**

I. Introduction

Bass *et al.* recognize in [1] that even a perfect architecture is useless if it is not understood; proper documentation should have enough detail, no ambiguity, and it must be organized such that users can quickly find information. Documentation of software architecture serves three important purposes: it is used for education, system analysis, and it is the primary vehicle for stakeholders communication [2]. Referring to architecture documentation, Kruchten suggests that if it is not documented, it does not exist [3], this is applicable to both the knowledge contents and the ways the knowledge is organized. In the software industry, it is common practice to share and retrieve architectural knowledge using file-based documents, diagrams, wiki systems, specialized tools and the like. This documentation approach has not changed for decades, however, it has various issues when retrieving Architectural Knowledge (AK):

- The way architects structure a document is reflected in its table of contents, which provides an index on the AK. If the readers search for AK which is outside this index, then the AK may not be easily found.
- Cross-referencing between different sections and documents, in order to make AK traceable, is difficult. Tables can help, but are hard to maintain.
- If structuring of document content is not done properly, AK can become redundant and scattered across architecture views, topics, and documents. This is hard to prevent when AK evolves and when there are many stakeholders with different needs for the AK.

These issues occur because file-based documents have a linear organization of contents. The limited support for structuring and indexing contents results in documents that provide a single-dimensional perspective on AK.

In this paper we describe an approach aimed at addressing these issues of file-based documentation. We follow up on the approach to architecture documentation proposed in [4] to implement a lightweight software ontology and a semantic wiki tool. We refer to such an approach to architecture documentation as an 'ontology-based' approach.

We report on a controlled industry experiment in which the ontology-based approach is compared to a file-based approach when software professionals retrieve AK. In this experiment, software professionals were asked to find knowledge about architectural elements such as subsystems, components, interfaces, behaviour, requirements, and decisions using both documentation approaches.

The experiment yields statistical significant evidence that the ontology-based approach to architecture documentation is more time-efficient for retrieving AK and in some cases more effective. The ontology-based approach provides for a classification of, and a multitude of relationships between, knowledge elements to quickly locate the AK one is looking for. The file-based documentation provides much less diverse AK retrieval possibilities. Moreover, the ontology-based approach forces one to be formal and consistent in labeling knowledge elements, which further improves AK retrieval efficiency and effectiveness.

In section II we provide the background on architecture documentation approaches and challenges, and architecture ontologies. In section III our ontology-based approach and related work are described. Section IV provides details on the experiment, section V on the findings, and section VI on a questionnaire for the experiment. Threats to validity are discussed in section VII. Section VIII reports our conclusions and future work.

II. Background - Architecture Documentation

A. Architecture Documentation Approaches and Challenges

Parnas [5] argues 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 architecture documentation structures can achieve this separation by using, for example, a view-based structure [1], [2], [6].

Each view provides a 'cross-section' of AK. View-based descriptions are useful to stakeholders who are interested in different cross-sections of the knowledge about a system. Cross-referencing of knowledge between views can ensure that interrelated relevant knowledge is 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 is recognized in [7], where the notion of perspectives is introduced next to that of views. Perspectives serve to organize 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 knowledge from file-based documentation. In practice, most stakeholder concerns are each addressed by a small documentation subset that is different for each concern [8]. Extensive use of cross-references between scattered knowledge or (alternatively) redundant recording of knowledge in file-based documents makes knowledge retrieval and maintenance impractical and error prone, especially when knowledge evolves. A number of AK retrieval challenges (adapted from [9]) exist.

1) *Architecture documentation understanding*
   Document understandability becomes more challenging when documentation size increases in large and complex systems [2]. Especially when stakeholders have different backgrounds or use of language. The original intention of the authors is often lost [9].
2) *Locating relevant architectural knowledge*
   Knowledge is often spread over multiple documents [10], and version mismatches (due to informal sharing) as well as the lack of fine granularity in documents make locating AK in architecture documents hard [9].
3) *Support for traceability between different entities*
   Providing traceability between documentation sources is difficult [11]. Text and tables are limited in communicating different relationships [9].
4) *Support for change impact analysis*
   Making and changing decisions or requirements impacts parts of the architecture. Because decisions and their relationships are usually not explicit, it is often very hard to reliably analyze and predict the impact of the changes [9].
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 [1], [12]. These qualities are inherent to the architecture per se and the size and complexity of architecture documentation directly influence qualities themselves as well as their assessment [9].
6) *Trust in the credibility of information*
   AK changes often in large and complex systems and the cost to update is sometimes prohibitive [9], especially in agile environments where change is fast and documentation has less focus than in waterfall environments. Documentation is quickly outdated and stakeholders lose confidence in its credibility [13].

B. Software Architecture Ontologies

"An ontology" refers to a formal domain model describing its concepts and relationships among concepts [14]. Ontologies enable a hierarchical classification of interrelated domain concepts and can for instance be represented using an RDF Schema and the more expressive 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 bring advances in knowledge management to web-based systems [15].

Several ontologies and domain models have been proposed in recent years for expressing AK as they can be used to capture, manage, and share architectural design decisions explicitly [16] as well as providing a common vocabulary and a level of precision needed for making architecture decisions [17].

III. Ontology-Based Documentation Approach

A. Lightweight Software Ontology

In this experiment, we use the lightweight software ontology from [4] for annotating knowledge in requirement and architecture documents. The lightweight software ontology is designed to be flexible so that it can be adapted for specific application domains. As intended in [4], the ontology was adapted to suit the terminology and taxonomy used in the industry experiment domain (see Figure 1).

We illustrate the ontology with a software development scenario (the classes are marked boldface and the semantic relationships are marked italic):

A software architect makes the **decision** that **non-functional requirement** 'configurability' should be *realized by* the **architecture**. The **decision** *results in* **behaviour** 'user preferences' which *satisfies* the **non-functional requirement** 'configurability' and a new **functional requirement** 'set user preference'. When a software engineer implements **behaviour** 'user preferences', s/he needs to know which **settings** can be *changed by* and *stored by* this **behaviour**. S/he also needs to know the **interfaces** that are necessary to *realize* the **behaviour** as well as details on the

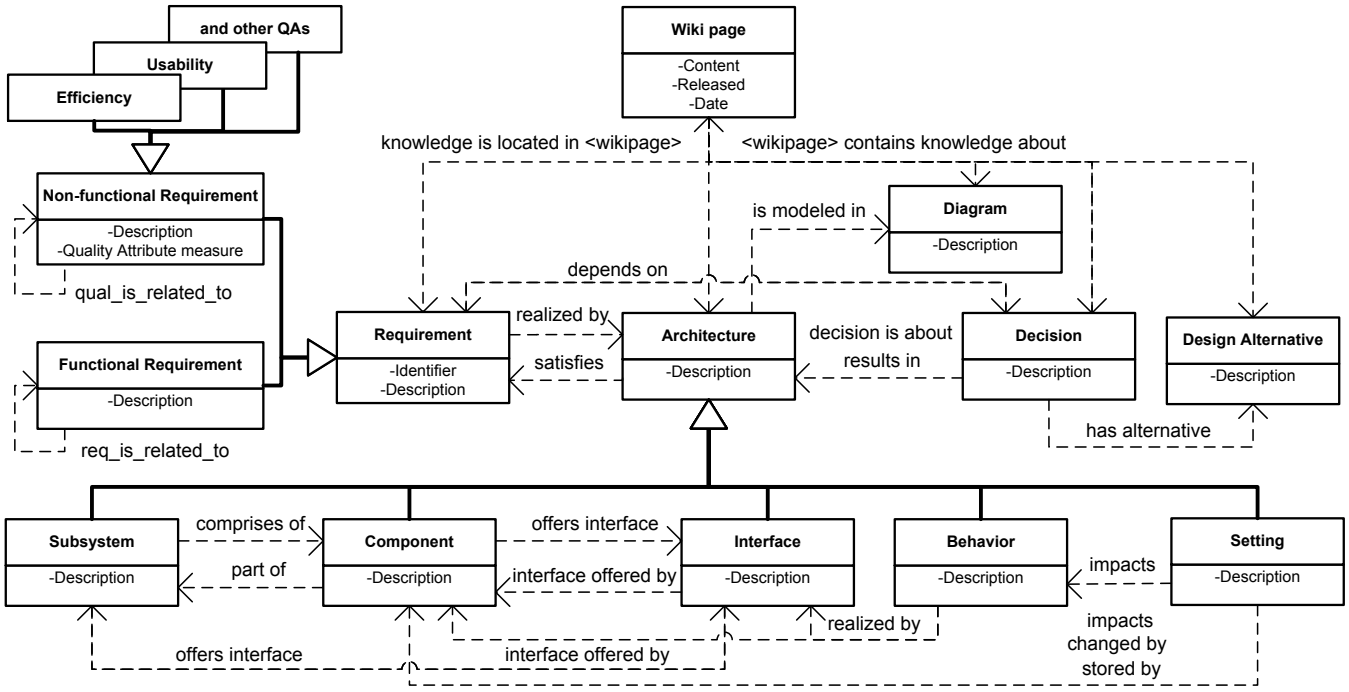

Figure 1. Software ontology adapted for Océ experiment domain.

components and **subsystems** that *offer* these **interfaces**. When implemented, the **behaviour** can be tested using the **requirements** that are *realized by* the **behaviour** and under various **settings** that *impact* it. Additional information about such an ontology can be found in [18].

The semantic relationships defined in the ontology and illustrated in the scenario above address challenge (3) *Traceability* and (4) *Change impact*. Checking the existence of semantic relationships, e.g., 'requirements and behaviour must be *realized by* architecture elements', addresses challenge (5) *Assessing design maturity*.

B. ArchiMind semantic wiki

We used the OntoWiki tool [19] as the basis for the tool in our ontology-based approach. OntoWiki allows for web-based visualization and management of (ontology and its instances in) knowledge bases in RDF. OntoWiki is similar to existing wiki systems (e.g. Wikipedia). In addition, it offers semantic-enhanced search facilities such as filtering, faceting, and graph-like exploration of knowledge. OntoWiki is open source and aims to support collaborative knowledge engineering. We based our choice for OntoWiki on evaluation of semantic wikis by Hoenderboom *et al.* [20] and Tamburri [21].

Adaptations were made to OntoWiki to enhance the system for input of architecture documentation and semantic annotation of documentation content. We named the adapted version 'ArchiMind'. File-based documentation from e.g., Word processors or e-mails, can be copied and pasted in a WYSIWYG editor and stored as wikipages. The length of copied documentation fragments, e.g., a paragraph, section, or chapter, is variable to allow for a coarse or fine granularity.

ArchiMind allows for semantic annotation of architectural concepts (the classes in Figure 1) in text on wikipages, described in detail in [22]. When a piece of text in wikipages is annotated, a semantic relationship is created from the text in wikipages to the AK instances in the ontology. The text on the wikipages is highlighted and, when it is clicked on, showing information about the architectural concept and its semantic relationships to other AK instances. The semantic annotation mechanism allows for locating (sources of) AK and addresses issues with synonyms, homonyms, spelling errors, abbreviations, ambiguity, and context-dependent interpretation in architecture documentation text. This addresses the challenges: (1) *Architecture documentation understanding* and (2) *Locating relevant architectural knowledge* (See section II-A).

Dublin Core [23] is used to describe documentation metadata; e.g., date and version of documentation sources describing AK give insight in whether they are up to date. Next to the native version control in OntoWiki, also basic version control of wikipages was implemented in ArchiMind. This partially addresses challenge (6) *Credibility of information*.

C. Related Work

Su *et al.* proposed KaitoroBase [24], an architecture documentation exploration and localization tool, built on freebase

semantic wiki, for visualization and non-linear navigation of SADs stored in wiki pages. A meta-model based on Architecture Driven Design is used, however, there are no clear details on whether the meta-model itself can be adapted in the system and if it supports other types of architecture documentation and concepts therein. Su *et al.* provide exploration from a single node (say a single requirement), whereas our approach allows exploration from a set of related nodes (say all requirements realized by a component).

López *et al.* [25] proposed the Toeska Rationale Extraction (TREx) approach to recover, represent, and explore design rationale in text documents, including in-page semantic annotation. Ontology-based representation of rationale and architecture concepts is used. Their focus on rationale as well as rationale knowledge exploration on HTML pages, missing the benefits of a semantic wiki, is a difference to our approach. Validation in a case study by measuring precision, recall, and effort in time is similar to our experiment setup, however, complete rationale recovery from an industry corpus by students is used, as opposed to a specific question set answered by industry professionals.

Jansen *et al.* [9] proposed a method and a tool which allows users to semantically enrich content of MS Word documents for increased architectural understanding and capture AK from MS Excel and Python programs. A separate knowledge explorer tool provides visualization of annotated AK instances and their relationships. Knowledge and documentation are represented in separate tools, as opposed to our use of an integrated semantic wiki tool. Validation was done on architectural understanding between enriched and non-enriched file-based documentation both viewed in MS Word.

Traditional wiki systems have been used in some software organizations for architecture documentation. The use of tags and categories can help in retrieving knowledge, however tags quickly lose meaning when used arbitrarily. Categories help in structuring knowledge but either are too static or have high maintenance cost. Hyperlinks provide cross-referencing but they lack the semantics that are necessary to express the meaning of the relationships between architecture elements.

IV. Experiment setup

The experiment was conducted at the R&D department of Océ, an international leader in digital document management and a Canon Group company. The architecture documentation corpus used in the experiment belongs to the software used in a series of document printing machines developed at Océ R&D.

The primary experimental goals are:

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

Participants in the experiment are all software professionals at Océ R&D who are involved in the software development process. Table 1 below gives the demographics of the experiment participants.

Initial recruiting of participants was done by presenting a voluntary sign-up list for the experiment at the end of a presentation on architecture documentation at Océ by one of the authors. The presentation was advertised using a mailing list and posters. Recruiting also took place at the end of each experiment by asking participants to recommend the experiment to interested colleagues.

At the start of each experiment we informed participants of confidentiality of their individual results to Océ and any researchers other than the experiment supervisor.

The materials used in the experiment consist of a software documentation corpus and questions about the AK in that corpus that are to be answered by experiment participants. The document corpus used in the experiment consists of 7 documents, 79 pages, 1794 paragraphs, 3183 lines, and 13,962 words in total.

The experiment document corpus is further specified into:

- Two Software Architecture Documents (SAD) of 3 and 9 pages, respectively. SADs detail the design of functionality, behaviour, and components. One SAD gives an overview of what knowledge can be found 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 SBD describes all possible settings for behaviour.
- One System Reference Document (Sysref), whose size is 19 pages. The Sysref document details the high level system design and decomposition, in terms of subsystems, components, and interfaces, and gives design decisions and rationale on system design.
- One Design Document that contains three UML diagrams. The diagrams detail the design of subsystems, components, and interfaces. The design document is often more up to date than the Sysref document that partially details the same knowledge.

These documents form a representative subset of the available types of documents. Three Océ software professionals verified and confirmed that the data set was representative of their usual practice. Question 6 in Table V, belonging to a questionnaire after each experiment session, confirms this.

The documents are a mixture of rather formal text, as in ”REQ123-Behaviour X must allow users to login using interface Y”, where requirements are numbered and prefixed with ”REQ”. At other places, descriptions of, say, settings and components involved in behaviour, are not easily recognized (their type is not clearly indicated, or their occurrence

Number of participants	Primary role of participants	Average years in role at Océ	Average years in role	Average years working at Océ
6	Domain architect	3.27	3.27	10.08
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 (via external agency)
1	Software designer	1	1	1

Table I
EXPERIMENT PARTICIPANTS DEMOGRAPHICS

is not clear from the document structure); we call the latter informal.

The documents were entered in ArchiMind as wikipages. We then identified and annotated 214 instances using the ontology presented in Figure 1, namely;

- 27 wikipages and 3 diagrams
- 45 functional- and no non-functional requirements
- 22 decisions and 3 alternative decisions
- 19 subsystems, 66 interfaces, and 15 components
- 8 settings and 6 behaviours

The annotations were verified by two software professionals from Océ during a pilot study, by asking whether specific AK instances were correctly classified (corresponding to an ontology class) and correctly interrelated by semantic relationships such as "requirement X is realized by component Y" and "decision X is about setting Y".

From the questionnaire (see also Section VI) we identified the main uses of documentation by our participants. Domain architects, workflow architects, project managers are the most intensive users of all types of architecture documentation. Software engineers are regular users of most types of documentation. The configuration manager uses SADs and SBDs and the software designer uses SBDs, SADs, and interface proposals. Product- and system test engineers are mostly interested in requirements, behaviour, and impact analysis but less interested in architecture and design as they view the software system as a black box.

We devised a question set for the experiment based on the annotated AK instances in the documents. We devised 4 question types and 7 concrete questions. The questions have been obfuscated for non-disclosure reasons: 'XX', 'YY', 'ZZ', and 'QQ' replace an actual software entity or concept. Two software professionals participated in a pilot experiment to evaluate, test, and improve the questions.

The questions used are the following:

- *1A: Which settings have an impact on behaviour "XX"?*
- *1B: Which settings have an impact on behaviour "YY"?*
- *2: Which requirements for behaviour "XX" should be satisfied (realized) by component "YY"?*
- *3A: Which decisions have been made around component "XX"?*
- *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?*

Experiment participants answered these questions using either the ontology-based approach or the file-based approach. They were asked to think aloud, verbally state their answers, and indicate when they were satisfied with the correctness of- and time spent on an answer. Participants were instructed that this satisfaction should reflect their normal way of working.

We formulated the following null and alternative hypothesis for experiment goal A and B;

H_{0A} = *There is no difference in time-efficiency between the use of the ontology-based and file-based approach for answering experiment questions.*

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_{0B} = *There is no difference in effectiveness between the use of the ontology-based and file-based approach for answering experiment questions.*

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.*

Two independent variables (or 'predictor variables') are used in the experiment, namely the file-based and the ontology-based approach to architecture 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 [26], is used for measuring effectiveness:

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

where *recall* is the proportion of relevant items retrieved from the total set of relevant items in a system and *precision* is the proportion of retrieved items that is relevant in a result set. The relevancy of items, or 'ground truth', was verified with two Océ professionals.

Using the Shapiro-Wilk and Kolmogorov-Smirnov tests, we found that our measurements from the experiment are not normally distributed. Therefore we applied the non-parametric Mann-Whitney-Wilcoxon test to the experiment data. One-tailed tests for each experiment question are reported in the next section. We report statistical significance at the 0.05 level.

We designed our experiment to be executed in 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 a tutorial on the use of ArchiMind, questions 1 and 2 to be answered with the ontology-based approach, and questions 3 and 4 to be answered with file-based documentation. Version 2 starts with questions 1 and 2 to be answered with file-based documentation approach, the ArchiMind tutorial, and questions 3 and 4 to be answered with the ontology-based approach. This design is aimed at reducing any biases that may arise from the selection of subjects in a treatment group and a control group.

Consecutive participants in the experiment were alternated between the two versions of the experiment. Because ArchiMind was new to the participants, a tutorial was used to train the participants. To prevent the participants getting preconceived ideas on how to formulate a query, the tutorial did not contain architecture documentation and knowledge used for answering questions posed in the experiment.

We chose to execute the experiment with each participant individually in a meeting room, because executing the experiment at the desks of participants would introduce distraction for them and entropy in the experiment.

V. Experiment Findings

In this section we report the statistical test results and the analysis of our experiment.

A. Knowledge Retrieval Efficiency

The average time in seconds, required for answering each question when using the ontology-based approach and file-based approach, as well as the difference in time between the two approaches, is listed in Table II.

All results, except those for question 4A, are statistically significant as can be seen from table III. Consequently, we reject the null hypothesis $\mathbf{H_{0A}}$ and accept the alternative hypothesis $\mathbf{H_{1A}}$ for all questions except 4A.

Question 1A and 1B can be answered from a SBD that details on settings for all behaviour. The settings for the relevant behaviour are also documented in two SBDs that detail on either behaviour XX or YY. Almost all participants used the latter SBDs for answering 1A and 1B and they did not use the settings document. The two SBDs did not provide structure for retrieving settings, e.g. a dedicated settings section, and settings were informally recorded. This made it hard for participants to find settings quickly. Question 1 can be answered in ArchiMind using the *impacts* semantic relationship.

Question 2 can be answered from one SBD which contains the requirements for behaviour XX. The components that realize requirements are explicitly mentioned in the requirements themselves, however many participants read the requirement text to verify whether a component is actually involved in realizing the requirement. The structure of the SBD does not give information on where to find the requirements that are specifically realized by component YY. Question 2 can be answered in ArchiMind using the semantic relationships *satisfies* and *realized by*.

Question 3A can be answered using the system reference document which contains some sections on rationale. These sections are however structured on a subsystem level and not on a component level. Participants would either have to know to which subsystem component XX belongs or use a keyword search. Keyword search is not very efficient as the component is mentioned multiple times throughout the document. Question 3B can be answered from one of the SBDs where it is explicitly described in an appendix. Various participants had trouble finding the decision quickly as it was only described in one of the behaviour documents, whilst the decision impacts behaviour described in three SBDs available in the experiment document corpus. Question 3 can be answered in ArchiMind using the semantic relation *decision is about*.

The answer to question 4A was quickly found by most participants using a keyword-search in the system reference document. Question 4B can be answered from the system reference document, however, most participants would also check the UML diagrams in the design document. This since they are aware that the latest changes are reflected in the UML diagrams and the system reference document could be outdated. The UML diagrams however depict the entire system consisting of 66 interfaces and 19 subsystems. This made it difficult for participants to locate the interface and subsystem found in question 4A as they would have to recognize the subsystem or interface visually from the diagram. Question 4 can be answered in ArchiMind using the semantic relationships *offers interface* and *interface offered by*.

Informal recording of AK and an unoptimized document structure for retrieving AK in the file-based documentation is a feasible explanation for the significant difference in time between the ontology-based and file-based approach for questions 1, 2, and 3. The presence of outdated information and unfamiliar diagram notations could explain the significant result for question 4B.

Question	1A	1B	2	3A	3B	4A	4B
Time ontology-based approach	161	157	229	148	197	73	40
Time file-based approach	394	212	382	401	374	78	64
Difference (in seconds)	233	55	153	253	178	5	24

Table II
AVERAGE TIME IN SECONDS REQUIRED PER QUESTION AND DIFFERENCE

Question	*p-value*
1A: *Which settings have an impact on Behaviour XX?*	0.0092
1B: *Which settings have an impact on Behaviour YY?*	0.0324
2: *Which requirements for Behaviour XX (should be) satisfied (realized) by component YY?*	0.0060
3A: *Which decisions have been made around component XX?*	0.0001
3B: *Which decisions have been made on the configuration of Behaviour XX, YY, ZZ and QQ?*	0.0013
4A: *Of which subsystem is interface XX part of?*	0.5510
4B: *Which other interfaces are offered by this subsystem?*	0.0156

Table III
STATISTICAL TEST RESULTS FOR TIME PER QUESTION

Question	*p-value*
1A: *Which settings have an impact on behaviour XX?*	0.3790
1B: *Which settings have an impact on behaviour YY?*	0.0306
2: *Which requirements for behaviour XX should be satisfied (realized) by component YY?*	0.0589
3A: *Which decisions have been made around component X?*	0.0112
3B: *Which decisions have been made on the configuration of behaviour XX, YY, ZZ, and QQ?*	0.0504
4A: *Of which subsystem is interface XX part of?*	0.1650
4B: *Which other interfaces are offered by this subsystem?*	0.0478

Table IV
STATISTICAL TEST RESULTS FOR $F1$-SCORES PER QUESTION

B. Knowledge Retrieval Effectiveness

A higher average $F1$ score was observed for all questions answered by the ontology-based approach in the experiment.

As can be seen from Table IV, the difference in $F1$ score between the ontology-based approach and file-based approach is statistically significant for questions 1B, 3A, and 4B, with p=0.0306, p=0.0112, and p=0.0478 respectively. Consequently, we reject the null hypothesis $\mathbf{H_{0B}}$ and accept the alternative $\mathbf{H_{1B}}$ for questions 1B, 3A, and 4B.

The difference in significance between questions 1A (insignificant with p=0.3790) and 1B (significant with p=0.0306) is interesting as these two questions are very similar. Both can be answered from SBDs and both have the same amount of correct answers, namely, two settings. We observed that the settings for question 1B were recorded in a more obscure and informal way, as compared to settings for question 1A. This seems a plausible explanation for why question 1B is statistically significant and question 1A is not. This could indicate that settings are not always documented consistently and explicit enough in SBDs and that this affects the effectiveness of knowledge retrieval from those documents.

The difference in significance between questions 3A (p=0.0112) and 3B (p=0.0504) is most likely caused by the structuring and informal notation in the file-based documents. Because rationale is structured by sections per subsystem in the system reference document, it is hard to find the decisions for a specific component. Also, it was not explicitly documented whether a decision was made about a component or whether this component was mentioned as a part of background information. The answer to question 3B was recorded in an appendix in one of three behaviour documents that contain the answer. This was the only document that contains an appendix on design rationale. Also the table of contents lists the appendix clearly as having design rationale. This seems a feasible explanation for why the answer to question 3B is retrieved relatively correctly by participants.

Question 4B was in some cases incorrectly answered due to misinterpretation of UML notations. Also, as the document corpus was from an active project in the development phase, the latest changes in the architecture are reflected in the UML diagrams and the system reference was outdated at some places. One participant was unaware of this and the others used the UML diagrams for verifying the correctness of the answer found in the system reference. Annotating these documents forced us to make it explicit in the semantic wiki whether AK was outdated between documents.

The answers to questions 2 and 4a were documented in a relatively formal and structured way in the file-based documentations, and therefore easy to retrieve.

The above evidence indicates that there is a significant difference in effectiveness between the ontology-based and file-based approach when AK must be retrieved that is informally documented or when the file-based document structure is not optimal for retrieving this AK. Also the presence of unfamiliar notations and outdated AK is an explanation for the difference in effectiveness between the two approaches for question 4.

VI. Questionnaire

After the experiment we asked each participant to fill in a questionnaire to collect qualitative evaluations on the file-based approach and the ontology-based approach (referred to as semantic wiki here). The questions, answers, and elaborations are reported in Table V.

The main source of software knowledge are 'colleagues' for most participants. Participants often stated during the experiment that they normally consult with their colleagues when documentation does not provide a clear answer.

VII. Threats to validity

We used [27] for designing the experiment and guidelines from [28] for reporting. In our experiment plan, we accounted for possible threats during the experiment design.

A. Construct validity

We recorded time to evaluate knowledge retrieval efficiency. Evaluating efficiency by the (intellectual) effort required [26] is much harder to do precisely and objectively. Time also largely represents the costs of using one of the approaches.

The $F1$ score is used to evaluate effectiveness as *"there is no absolute sense in which one can say that one particular pair of precision-recall values is better or worse than some other pair, or, for that matter, that they are comparable at all"* [26]. Retrieval of an irrelevant AK (precision) and lack of relevant AK (recall) is considered as equally ineffective when evaluating experiment results.

B. Internal validity

The role distribution of experiment participants in Table I shows that a population selection bias based on role is unlikely. On average, our participants have worked in their role for 5.75 years and have been employed for almost 12 years at Océ. This gives confidence that experiment participants are familiar with documentation approaches at Océ. Participants who signed up for the experiment during the presentation might have introduced a selection bias as they might have a positive attitude towards any documentation approach other than the current approach used at Océ.

A subset of the documentation from one particular project was used. Participants from that project had, more than other participants, prior knowledge of this documentation. However, the type of documentation used as well as the type of questions asked were generic. To avoid any bias, participants were not informed about how much and precisely which documentation was present in the subset and they were instructed to always find and verify answers in the experiment documentation despite any prior knowledge. Participants would first have to verify prior knowledge as it might not be in the used documentation subset.

The questions asked during the experiment are all supported by the software ontology, i.e. the questions posed in the experiment are about AK that corresponds to the classes and relations defined in the ontology. This limits our results as they cannot be replicated when AK needs to be retrieved that is unsupported by the ontology or when the class and type of this AK are unknown.

We took care to make sure that ArchiMind did not contain more or less architectural information, content wise, than is in the file-based documentation against which it is compared.

In the document corpus provided for the 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. Questionnaire results also confirm this. We chose to not update any AK in order for the experiment document corpus to be realistic.

C. External validity

The experiment was conducted at one company which means the findings are specific and cannot be generalized beyond this study. The specific set of questions asked in the experiment also limits generalization. The use of SADs, SBDs, system reference-, and design documents can be considered generic documentation practice in industry.

D. Conclusion validity

As the data collected in the experiment is not normally distributed, we cannot calculate statistical power under the assumption of normal distribution. The increased chance of a Type II error (false negative), when using a non-parametric test on normally distributed data, does not apply here [29].

1	*When searching for software knowledge, would you evaluate the semantic wiki, as compared to using normal documentation, as:* Better - 24 (92.3%) Worse - 0 (0%) Making no difference - 2 (7.7%)
Comments:	Most participants find (semantic) relationships important and useful when searching software knowledge. Also the provided search mechanisms, facets, structure, and (centralized) accessibility are elaborations given by participants.
2	*Do you experience troubles in your daily job at Océ when searching for software knowledge using the standard documents?* [1] Yes - 23 (88.5%) No - 3 (11.5%)
Comments:	Most participants mention 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.
3	*Do you think that the semantic wiki can provide you with better search mechanisms than you currently have at your disposal?* Yes - 25 (96.2%) No - 0 (0%) I do not know - 0 (0%) No opinion on this - 1 (3.8%)
Comments:	Most participants mention that the semantic relationships are useful for searching.
4	*Do you think it's worthwhile to set up a semantic wiki at Océ for searching software knowledge & documentation management?* Yes - 17.5 (67.3%) No - 2 (7.7%) I do not know - 6.5 (25%) No opinion on this - 0 (0%)
Comments:	Most participants comment they do not know whether the benefits of the ontology-based approach outweighs the costs. Other elaborations given are that enough effort should be invested, authorization should not be an obstacle, training should be provided and that the knowledge in the system should be complete, maintained well and reviewed by an expert. One participant chose both options 'yes' (for management) and 'I do not know' (for searching).
5	*From which sources do you normally get knowledge about the software made at* Océ?
Answers:	Most often mentioned are colleagues and after that documents, source code, Sharepoint, Docfinder, and CM Synergy.
6	*From which types of documents do you normally get knowledge about the software made at Océ?*
Answers:	Most participants use SBDs. Also SADs, interface- and functional specifications, diagrams, technical reports and source code are used as well as impact analysis-, high level architecture-, system reference-, and module design documents.
7	*What percentage of your time do you daily spend on searching and retrieving software knowledge?*
Answers:	19.75%. The answers range from 0% to '50% or more'. This question was answered by half of the participants

Table V
QUESTIONNAIRE RESULTS

VIII. CONCLUSIONS AND FUTURE WORK

The major contribution of this work is the empirical evidence to demonstrate that the use of an ontology-based approach to architecture documentation is more efficient and effective in retrieving AK than the use of a file-based approach. The document structure reflects a single-dimensional perspective that is inherent to file-based documents, and that limits efficient and effective knowledge retrieval. Moreover, the ontology-based approach forces one to be more formal when annotating relevant knowledge elements (such as component, interface, behaviour), which further improves retrieval efficiency and effectiveness. The limitations of AK structuring and description in file-based documents can thus be overcome by the ontology-based approach.

The participants indicated that they often rely on their colleagues for knowledge retrieval, especially when they are not satisfied with results from the available documentation. Many participants evaluate the ontology-based approach as better for retrieving knowledge than the file-based approach.

Though the results of this study have indicated that ontology-based AKM approaches hold a promising future, there are yet many more challenges to overcome. We plan to investigate into semi-automatic input (annotation) of knowledge using NLP, parsing, and form-based user input. Knowledge representation for personalization, as well as knowledge communication using some Push & Pull mechanisms will also be areas of investigations. Comparison between ontology-based- and hypertext-based approaches to software architecture documentation will be future work.

ACKNOWLEDGEMENTS

The authors wish to thank 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 their adaptations to OntoWiki. This research has been partially sponsored by the Dutch "Regeling Kenniswerkers", project KWR09164, "Stephenson: Architecture knowledge sharing practices in software product lines for print systems" and by the Natural Science Foundation of China (NSFC) project No. 61170025 "KeSRAD: Knowledge-enabled Software Requirements to Architecture Documentation".

[1] 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.

REFERENCES

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Addison-Wesley, 2003.

[2] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2002.

[3] P. Kruchten, "Documentation of software architecture from a knowledge management perspective - design representation," in *Software Architecture Knowledge Management*. Springer, 2009, pp. 39–57.

[4] A. Tang, P. Liang, and H. van Vliet, "Software architecture documentation: The road ahead," in *Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture WICSA '11*. IEEE Computer Society, 2011, pp. 252–255.

[5] D. Parnas and P. Clements, "A rational design process: How and why to fake it," in *Formal Methods and Software Development*. Springer LNCS, 1985, vol. 186, pp. 80–100.

[6] "IEEE recommended practice for architectural description of software-intensive systems," *IEEE Std 1471-2000*, 2000.

[7] N. Rozanski and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2005.

[8] H. Koning and H. van Vliet, "Real-life it architecture design reports and their relation to ieee std 1471 stakeholders and concerns," *Automated Software Engg.*, vol. 13, no. 2, pp. 201–223, April 2006.

[9] A. Jansen, P. Avgeriou, and J. S. van der Ven, "Enriching software architecture documentation," *J. Syst. Softw.*, vol. 82, no. 8, pp. 1232–1248, August 2009.

[10] R. C. de Boer and H. van Vliet, "Architectural knowledge discovery with latent semantic analysis: Constructing a reading guide for software product audits," *J. Syst. Softw.*, vol. 81, no. 9, pp. 1456–1469, September 2008.

[11] C. Hofmeister, R. Nord, and D. Soni, *Applied software architecture*. Addison-Wesley, 2000.

[12] J. S. van der Ven, A. Jansen, P. Avgeriou, and D. K. Hammer, "Using architectural decisions," in *Proceedings of the 2nd International Conference on the Quality of Software Architectures (QoSA)*. Karlsruhe University Press, 2006.

[13] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE Softw.*, vol. 20, no. 6, pp. 35–39, November 2003.

[14] C. López, P. Inostroza, L. M. Cysneiros, and H. Astudillo, "Visualization and comparison of architecture rationale with semantic web technologies," *J. Syst. Softw.*, vol. 82, pp. 1198–1210, August 2009.

[15] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*, 2nd ed. MIT Press, 2008.

[16] M. Shahin, P. Liang, and M. Khayyambashi, "Architectural design decision: Existing models and tools," in *Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture WICSA'09*. IEEE Computer Society, 2009, pp. 293–296.

[17] A. Akerman and J. Tyree, "Using ontology to support development of software architectures," *IBM Syst. J.*, vol. 45, pp. 813–825, October 2006.

[18] A. Tang, P. Liang, V. Clerc, and H. van Vliet, "Supporting co-evolving architectural requirements and design through traceability and reasoning," in *Relating Software Requirements to Software Architecture*. Springer, 2011, pp. 59 – 85.

[19] S. Auer, S. Dietzold, and T. Riechert, "Ontowiki a tool for social, semantic collaboration," in *5th International Semantic Web Conference ISWC2006*. Springer LNCS, 2006, vol. 4273, pp. 736–749.

[20] B. Hoenderboom and P. Liang, "A survey of semantic wikis for requirements engineering," SEARCH, University of Groningen, Tech. Rep., 2009.

[21] D. A. Tamburri, "An architecture description viewpoint wiki based on the semantic web paradigm," Master's thesis, 2010.

[22] K. A. de Graaf, "Annotating software documentation in semantic wikis," in *Proceedings of the fourth workshop on Exploiting semantic annotations in information retrieval ESAIR'11*. ACM, 2011, pp. 5–6.

[23] J. Kunze and T. Baker, "Dublin core metadata element set, version 1.1," Internet Engineering Task Force, Tech. Rep. RFC 5013, 2007.

[24] M. T. Su, C. Hirsch, and J. Hosking, "Kaitorobase: Visual exploration of software architecture documents," in *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering ASE'09*. IEEE Computer Society, 2009, pp. 657–659.

[25] C. López, V. Codocedo, H. Astudillo, and L. M. Cysneiros, "Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach," *Science of Computer Programming*, vol. 77, no. 1, pp. 66–80, January 2012.

[26] C. van Rijsbergen, *Information Retrieval*, 2nd ed. Butterworths & Co, 1979.

[27] F. Shull, J. Singer, and D. I. Sjøberg, *Guide to Advanced Empirical Software Engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[28] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, "Reporting Experiments in Software Engineering," in *Guide to Advanced Empirical Software Engineering*. Springer London, 2008, ch. 8, pp. 201–228.

[29] A. Field, *Discovering Statistics Using SPSS - 2nd Edition*. Sage Publications, 2005.