In Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture, 20-24 June 2011 Boulder, Colorado, USA (c) IEEE 2011

Capturing architecture documentation navigation trails for content chunking and sharing

Moon Ting Su, John Hosking
Department of Computer Science
The University of Auckland
Auckland, New Zealand
e-mail: {msu010|john}@aucklanduni.ac.nz

John Grundy
Faculty of Information & Communication Technologies
Swinburne University of Technology
Victoria, Australia
e-mail: jgrundy@swin.edu.au

***Abstract*—Navigating and understanding complex software architecture documentation is often challenging. To support finding relevant information in architecture documents (ADs), we propose a semi-automated approach based on the actual usage of ADs by previous users, i.e. by capturing users' exploration paths through ADs and making these paths available for future retracing and analysis. To do this, we have built a prototype tool (KaitoroCap) that captures users' AD exploration paths and saves them with contextual metadata. KaitoroCap displays the exploration paths in hierarchical tree-views and these exploration paths can be searched. This is helpful for recalling previous navigations and to follow others' useful paths in finding relevant information in AD. Our approach also enables dynamic restructuring of ADs and incorporates user rating, tagging and commenting of the content of ADs. Initial user evaluation shows promising results.**

Keywords – navigation; exploration; capture; software architecture document; documentation; architectural knowledge management

I. INTRODUCTION

The difficulty of finding useful content in overwhelming amounts of information is a key problem of software documentation [1]. For software architecture (SA) documents (ADs), this problem is further aggravated by the various stakeholders having only partial interest in the content of the ADs i.e. typically only limited content is useful for each stakeholder, sometimes as little as 25% of the AD [2]. For example, a developer looking for information in order to modify the code base is often interested in some similar, but much different, information to a systems administrator looking to deploy and manage the software.

To support finding relevant AD information, we argue that the architectural information in an AD needs to be structured into or presented as *chunks* [3]: groups of related pieces of architectural information. Chunks enable architectural information seen as being related by a body of users, but which otherwise might be dispersed, to be retrieved collectively as a unit, thus simplifying search by new users. To further this idea, we propose a new semi-automated approach based on the actual usage of ADs by previous users, i.e. by capturing users' exploration paths through the documents and making these paths available for future retracing and analysis. A user's exploration path through the sections of an AD indirectly links related information throughout the document together. This allows them to be retrieved collectively as a unit, making searching for related information easier. A path serves as a rudimentary chunk. However, additional analysis needs to be performed on the navigation path data to discover the real chunks.

This paper focuses on the results of a user evaluation of a prototype tool (KaitoroCap) that we have built as a proof-of-concept. KaitoroCap provides functionality to capture users' exploration paths through documents uploaded as wiki pages in Atlassian's Confluence Wiki [4]. We describe related work, our approach, features of KaitoroCap and our preliminary user evaluation results, discussion, conclusion and future work.

II. RELATED WORK

Views & Beyond suggests the use of documentation roadmaps and view templates to help new stakeholders to find the information that they need in an AD [5]. The former introduce readers to the organization of an AD. As a standard organization for a view, the latter help readers to find related sections. Concept maps are also used to show main concepts of an AD improving its readability [2]. In neither approach, have reports of benefit based on actual use been presented. Latent Semantic Analysis (LSA) has been used to construct suggested documentation readings to guide software product auditors to find required Architectural Knowledge (AK) [6]. LSA uses association of terms with documents ("semantic structure") to detect relevant documents based on searched terms [7]. Our study differs by helping readers find relevant information by analyzing traces of actual explorations of the AD by previous readers to suggest how it should be read.

Access to and delivery of the right AK in a Just-in-Time manner requires effective and lightweight search, including finding relevant information *within* documents [8]. In our study, relevant information inside documents is retrieved collectively as a unit i.e. path, making search for relevant information easier. AK management tools are reviewed in [9, 10]. None capture user exploration of architectural information nor do any support finding collective information based on the actual usage patterns of end users.

Read wear captures the reading history of documents [11], implemented by accumulating the read time for each visible line. Our approach calculates read time per section and not per line, as we believe individual lines are too low a level for chunking of architectural information. Team Tracks [12] shows source code navigation patterns of team members using techniques such as Favourite Classes (visit frequency)

and Related Items (visit sequence). Our work differs by focusing on *navigation* patterns of software AD.

On task activation, Mylar/Mylyn [13] monitors user interaction with code to form a task context as a degree-of-interest (DOI) model [13]. This filters the Eclipse interface to show only essential program elements related to the task based on the elements' interaction frequencies. We also consider the order of visitation of elements and analyze interaction data across different users to find common usage patterns. We focus on semi structured ADs whereas Mylyn deals with structured program elements. Recent work on degree-of-knowledge [14] links DOI with degree-of-authorship. It spans multiple tasks but still per developer. Mylyn's commercial extension, Tasktop [15], extends task contexts to documents and web sites. It doesn't inspect sections in a document, but does extract web hyperlinks to present as Navigator sub-nodes.

Our exploration paths bear similarity to the workflow in VisTrails [16]. Both capture a history of exploratory task steps. KaitoroCap doesn't support path editing e.g. changing the navigation sequences, or the content of the sections in the exploration paths. Unlike VisTrails, where changes to the execution parameters of workflows produce visible new results, the usefulness of edited exploration paths is not obvious without further validation for KaitoroCap.

III. APPROACH

KaitoroCap captures and saves users' navigation paths through AD and their meta-data (providing contextual information about tasks) while undertaking goal directed explorations of architectural information in ADs (Fig. 1). It displays the exploration paths in hierarchical elidable tree-views. The exploration paths can be searched and retraced. KaitoroCap provides the option to display the content of the visited sections inside a path's tree-view visualization effectively making the path a restructuring of the AD, dynamically grouping related information. Our approach also allows user rating (R), tagging (T) and commenting(C) of AD content. Fig. 2 shows KaitoroCap in use browsing AD. This includes Creating a new AD page (A, B), and browsing pages by clicking on hyperlinks (C). The prototype automatically inserts RTC, expand (read more)/collapse features into each section of the opened page (D). These allow the user to rate, tag, comment, click on "read more" to expand a section and later collapse it (E). Whenever the user moves the mouse pointer into a section, a border surrounding the section is drawn to highlight that the section is the current focus of those interactions.

KaitoroCap is implemented as a plugin for Atlassian's Confluence Wiki [4]. Using a Wiki for collaboration and knowledge sharing is appealing for SA documentation [17] , [18]. Studies show wiki-based ADs support better document navigation [19]. Architecture documents are structured as a set of linked short web pages with deeper structure. However, the structures of wiki ADs are determined by AD producers rather than consumers. Our work aims to gain insight on the structuring of ADs based on actual usage by consumers. The 'structuredness' of wiki ADs is good for overview, but poses more difficulty in finding finer details [19]. We aim to mitigate this by using navigation paths to find chunks of collective information.

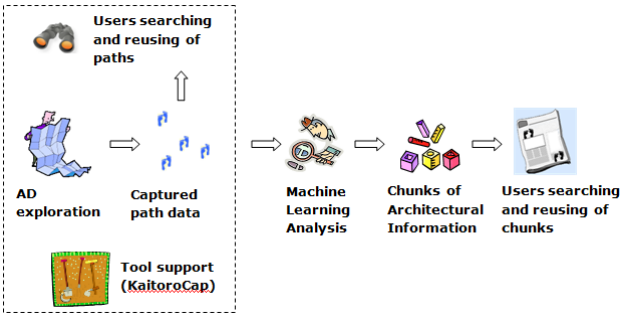

Figure 1. Exploration capture and chunking of architectural information .

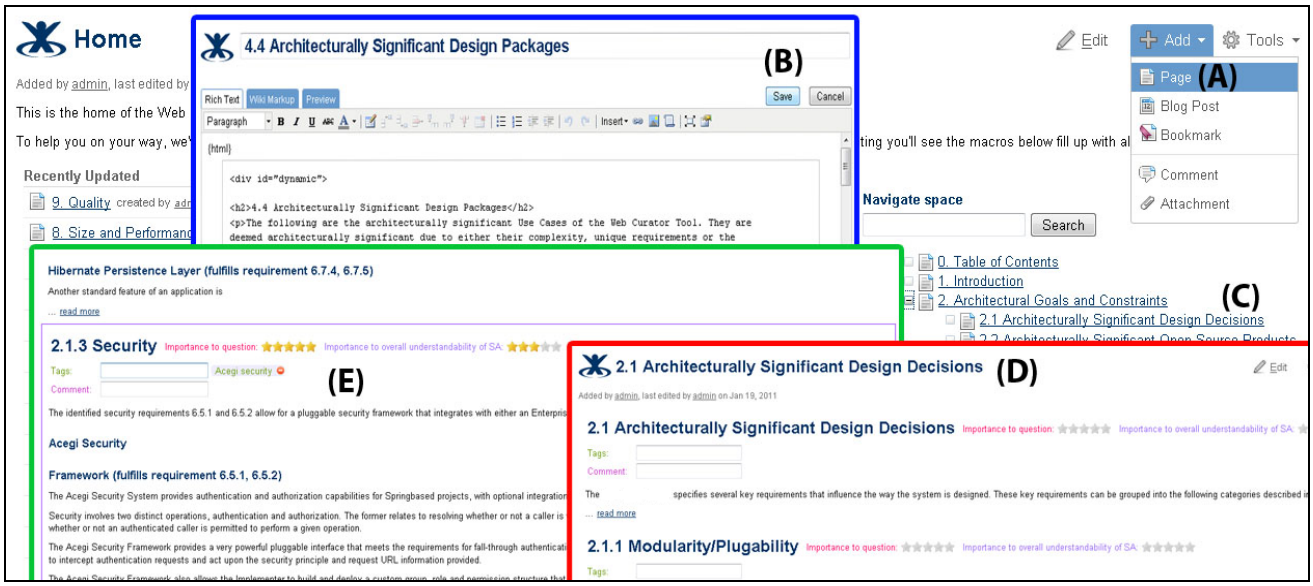

Figure 2. Example of KaitoroCap in use to explore and capture navigation traces.

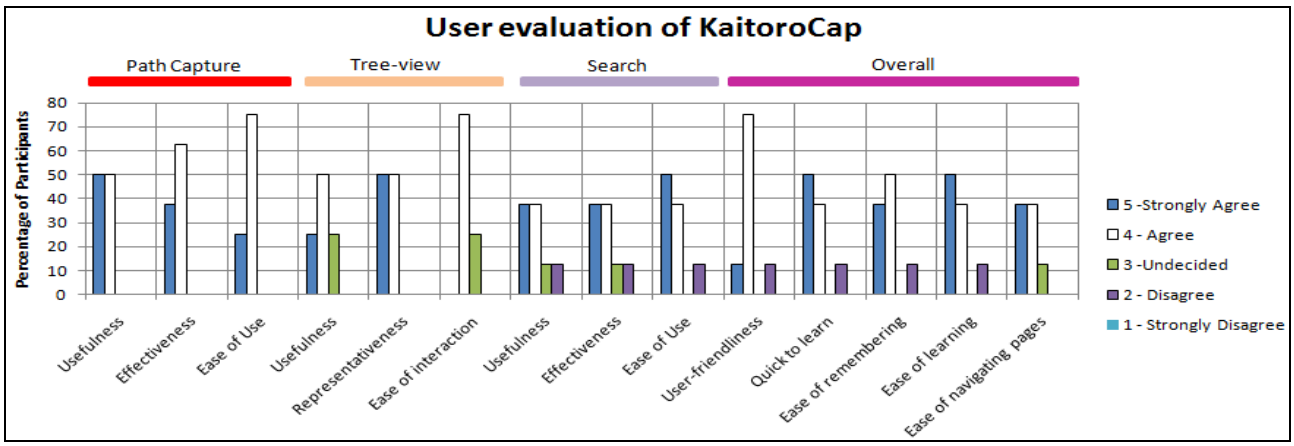

Figure 3. User evaluation of KaitoroCap

IV. User Evaluation Of KaitoroCap

We have conducted an initial user evaluation of KaitoroCap's features for exploration path capture, path search and the tree-view's suitability to visually represent paths. We recruited eight participants, who are PhD students or post-docs. None were directly involved in the design or implementation of KaitoroCap. All are experienced software developers. Background information gathered from them included level of education, English language proficiency, Wiki and AD experience, and domain knowledge related to an example AD domain. Participants explored this AD in KaitoroCap answering several provided questions relating to: overall software architecture of the system; how to change a specific part of the system; and how a quality attribute is realised. For each question, participants captured their own exploration paths using KaitoroCap's path-capturing feature. Participants' answers were captured and they were asked to rate AD sections they visited in terms of their importance to answering the question and the importance of the sections to their overall understanding of the software architecture of the system. They were also encouraged to tag/comment on visited sections. Participants we then surveyed to explore their perceptions of KaitoroCap's features.

The AD chosen for the evaluation study is a 24-page document defining the architecture of a real industrial system that manages digital web content for future preservation. Participants agreed or strongly agreed that the prototype is useful, effective and easy to use for path capture (Fig. 3). The tree-view visualization was rated only slightly little less positively with some undecided on its usefulness and ease of interaction. The search path feature had more spread in perception, but results were still very positive. Overall usability of KaitoroCap across all features was also assessed. In all cases only one participant rated characteristics less than a 4 on the 5-point Likert scale. Participants were queried on weaknesses and improvements. Among suggestions were: expanding the capture to include having search queries and text copied and pasted into the path; listing only sections visited for a faster view; providing more interesting visualizations, such as a timeline showing where most time is spent; keyword assistant for search; and widening the scope of search to include tags and comments. Overall, results show significant promise. Comments mainly suggested enhancements rather than questioning the core rationale of KaitoroCap. The current visualization approach is one area that merits such enhancement. One possibility is to construct a path visualization similar to KaitoroBase [20].

There are several threats to validity of our study. The way the AD was structured in the Wiki might influence participants' rating of KaitoroCap's path capture features. Alternative structuring of AD can be used to mitigate this. The 90-minute evaluation might affect a participant's focus and perceptions of KaitoroCap as they tire, but none reported negatively on this. Wrt external and conclusion validity threat, our (small number of) participants are from the same department and therefore unrepresentative. We are including participants from other institutions and industry in ongoing evaluations. For construct validity, a higher level of participant experience in accessing Wiki materials has a tendency to give more positive evaluation results. In our evaluation 87.5% of them have average experience in accessing materials in wiki environment but not in the Confluence Wiki used. Other threats relate to the next stage of this study dealing with analysis of the navigation path data to discover chunks; this will be reported in subsequent work.

V. Discussion, Conclusion And Future Work

While results are promising, we need more path data and better analysis of it to complete our vision. We will improve our prototype based on the evaluation results and build up a more substantial database of explored paths. Having captured more and varied paths, the correctness of answers given by participants will be validated to determine whether to include paths for further analysis. Validated navigation path data will be grouped by task and machine learning used to find common navigation patterns. These will serve as candidate chunks of SA information. Each chunk will be inspected to find its degree of completeness (comprising all related information needed for the task at hand), and whether the sequence of organization of its elements supports the understanding of the information in the chunk. This will provide one method of evaluation of the chunks. The chunks will also be included into KaitoroCap as recommended

exploration path(s) for the specific questions. Another study will validate end user perception of the chunks' usefulness.

A disadvantage of deducing user's actions by capture and analysis of path data is the difficulty of determining portions generated by non-intentional actions, e.g., the user may navigate absent-mindedly, but interactions pertaining to this are captured as with other interactions. We use the RTC features to help explicitly capture useful information, but this requires end user action to indicate relevance. It is inevitable that different users may provide contradictory ratings, tags and comments for the same content visited during similar information seeking task. Nevertheless, the averaged ratings as well as tags and comments for the elements (content) within a chunk provide an extra dimension to interpret the usefulness of the chunk.

Tags and comments can be displayed with sections, providing rich user-annotation of an AD. Sections with significant contradictory perceptions may flag problems in content. Tags and comments also afford other searching modalities as would use of semantic techniques. Following others' exploration paths doesn't guarantee better effectiveness or efficiency, but retracing navigation sequences of others', a newcomer can gain insight into their exploration processes enhancing task understanding. This is particularly so if paths are provided by more experienced co-workers. Our approach aims to enable user-driven chunking of SA information based on actual AD usage instead of perceived use. This allows closer match between authors' intentions and reader expectations, an important factor that determines the effectiveness of the documentation [21].

ACKNOWLEDGMENT

We thank the Ministry of Higher Education, Malaysia; PReSS, University of Auckland; and FRST Software Process and Product Improvement project for funding this research.

REFERENCES

[1] T.C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: the state of the practice", IEEE SOFTWARE, vol. 20, no. 6, 2003, pp. 35-39.

[2] H. Koning, and H. van Vliet, "Real-life IT architecture design reports and their relation to IEEE Std 1471 stakeholders and concerns", ASE, vol. 13, no. 2, 2006, pp. 201-223.

[3] M.T. Su, "Capturing exploration to improve software architecture documentation", Proc. 4th European Conference on Software Architecture: Companion Volume, ACM, 2010, pp. 17-21.

[4] "Confluence Wiki"; http://www.atlassian.com/software/confluence/.

[5] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford, Documenting software architectures: views and beyond, Addison-Wesley Professional, 2010.

[6] 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, 2008, pp. 1456-1469.

[7] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman, "Indexing by latent semantic analysis", Journal of the American society for information science, vol. 41, no. 6, 1990, pp. 391-407.

[8] R. Farenhorst, and H. Van Vliet, "Understanding how to support architects in sharing knowledge", Proc. 2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge, IEEE Computer Society, 2009, pp. 17-24.

[9] R. Farenhorst, P. Lago, and H. van Vliet, "Effective Tool Support for Architectural Knowledge Sharing", Software Architecture, LNCS, vol. 4758, 2007, pp. 123-138.

[10] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M.A. Babar, "A comparative study of architecture knowledge management tools", J. Syst. Softw., vol. 83, no. 3, 2009, pp. 352-370.

[11] C.H. William, D.H. James, W. Dave, and M. Tim, "Edit wear and read wear", Proc. SIGCHI conference on Human factors in computing systems, ACM, 1992, pp. 3-9.

[12] R. DeLine, M. Czerwinski, and G. Robertson, "Easing program comprehension by sharing navigation data", Proc. 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05), 2005, pp. 241-248.

[13] K. Mik, and C.M. Gail, "Using task context to improve programmer productivity", Proc. 14th ACM SIGSOFT international symposium on Foundations of software engineering, ACM, 2006, pp. 1-11.

[14] F. Thomas, O. Jingwen, C.M. Gail, and M.-H. Emerson, "A degree-of-knowledge model to capture source code familiarity", Proc. 32nd ACM/IEEE ICSE, ACM, 2010, pp. 385-394.

[15] R. Elves, "Tasktop for Eclipse - Get More out of Mylyn", 2010; http://tasktop.com/resources/tutorials/MoreOutOfMylyn.php.

[16] L. Moreau, I. Foster, J. Freire, C. Silva, S. Callahan, E. Santos, C. Scheidegger, and H. Vo, "Managing Rapidly-Evolving Scientific Workflows", Provenance and Annotation of Data, LNCS, vol. 4145, 2006, pp. 10-18.

[17] P. Louridas, "Using Wikis in Software Development", IEEE SOFTWARE, vol. 23, no. 2, 2006, pp. 88-91.

[18] R. Farenhorst, and H. van Vliet, "Experiences with a Wiki to Support Architectural Knowledge Sharing", Proc. Wikis4SE @ WikiSym 2008, 2008.

[19] F. Bachmann, and P. Merson, "Experience Using the Web-Based Tool Wiki for Architecture Documentation. ", Technical Note CMU, 2005.

[20] M.T. Su, C. Hirsch, and J. Hosking, "KaitoroBase: Visual Exploration of Software Architecture Documents", Proc. 24th IEEE/ACM International Conference on ASE 2009, pp. 653-655.

[21] R.C. de Boer, and H. van Vliet, "Writing and Reading Software Documentation: How the development process may affect understanding", Proc. 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, IEEE, 2009, pp. 40 - 47.