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

KaitoroCap : a document navigation capture and visualisation tool

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

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

***Abstract* — To facilitate the usage of software architecture documents (ADs), we claim the architectural information in the ADs needs to be structured into or presented as *chunks*. A chunk allows related information to be retrieved collectively as a unit and simplifies information location tasks. 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 ADs while engaging in information seeking tasks and making these paths available for future retracing and analysis. As part of our work, we developed KaitoroCap, a document navigation capture and visualisation tool. Its main features are exploration paths capture, retrieval, analysis, hierarchical tree-view visualization of paths, path searching, section rating, tagging, commenting, expanding/collapsing and page model generation to enable dynamic restructuring of ADs. This paper describes the design, implementation and usage examples of KaitoroCap.**

Keywords: navigation; exploration; capturing tool; software architecture document

I. INTRODUCTION

To facilitate the usage of software architecture documents (ADs), the architectural information in the ADs needs to be structured into, or presented as, *chunks* [1]. A chunk refers to a grouping of related information in the document. By pulling together related information which otherwise may be dispersed, a chunk allows related information to be retrieved collectively as a unit and simplifies information location tasks.

We propose a new semi-automated approach capturing users' exploration paths through ADs while engaging in information seeking and navigation tasks. The chunking of architectural information based on exploration paths is based on the notion that an exploration path indirectly links together that information throughout the AD that the user perceives may be related to the task at hand. Hence a path serves as a rudimentary invalidated chunk. To discover the real chunks, we will analyze substantial navigation path data with machine learning approaches to find common navigation patterns which serve as potential chunks. The direct involvement of the users of the ADs in this approach enables user-driven chunking of architectural information based on actual usage of the ADs instead of perceived usage.

Our approach incorporates user rating (R), tagging (T) and commenting (C) of the elements (content visited by the users during the exploration session) within an exploration path. The user ratings are to indicate the importance of the elements to an information-seeking task and their importance to the user's overall understandability of the software architecture of the system described by the AD. These context dependent actions enable more informative analysis of the exploration data and interpretation of the usefulness of the content of the AD and the potentially discovered chunks, in addition to analysis based on the frequencies of visit, time spent and the sequences of visitation. This paper focuses on the design, implementation and a usage example of KaitoroCap, our tool we have built as a proof-of-concept to capture and visualize users' exploration paths in documents uploaded as wiki pages in Atlassian's Confluence Wiki [2].

II. RELATED WORK

Several approaches have been proposed to help stakeholders to find the information that they need in an AD: documentation roadmaps [3], view templates [3], concept maps [4] and the use of semantic structure to detect relevant documents based on search term [5]. We analyze traces of previous readers' actual exploration of the AD to suggest how information should be chunked for faster access. Existing Architectural Knowledge (AK) management tools [6, 7] do not support the capture of users' exploration of architectural information, nor do they support finding collective information based on actual usage patterns. As a document exploration capturing tool, KaitoroCap differs from other tools that capture users' interaction information, such as Team Tracks [8] and Mylar/Mylyn [9] that capture users' interaction with source code, and VisTrails [10] that captures scientific workflows.

We also differ from existing work that analyzes exploration path data to find common navigation patterns. Instead of accumulating read time for each visible line as in read wear [11], we calculate read time per section (which can include paragraph, image, table, etc) as we believe individual lines are too low a level for chunking AD information. Apart from the interaction frequency used to determine the relevancy of program element in Mylar/Mylyn [9], we also consider the visitation sequences of document sections. Recent work on degree-of-knowledge [12] of Mylar/Mylyn combines it with degree-of-authorship. It spans multiple tasks but still per developer, whereas we analyze interaction data across different users to find common usage patterns. Tasktop [13] extends task contexts to documents and web sites but does not inspect sections in a document, though it does extract web hyperlinks to present them as sub-nodes in its Navigator.

III. KAITOROCAP

KaitoroCap captures users' exploration paths through a document and saves them with metadata to provide contextual information about the exploration. During the exploration, it allows section rating, tagging, commenting, and expanding/collapsing. It also provides the ability to visualize exploration paths as hierarchical tree-views that are collapsible and expandable to provide both succinct and detailed views of the paths. The content of visited sections is extracted and embedded inside a tree-view visualization of an exploration path, effectively a restructuring of the AD.

Fig.1 shows the high-level design of KaitoroCap. It consists of two main modules: Authoring and Exploration. The Authoring module comprises the AD Modeller, Template Maker and AD Authoring sub-modules. The AD Modeller is used to model the meta-model of ADs if it is required. The meta-model is saved and can be fed into the Template Maker to generate templates for ADs. The AD Authoring sub-module provides functions to create and edit the actual documents which are saved in the repository. It is also responsible for automatically creating a page model for each page created or edited. The page model improves performance by minimizing the amount of detail saved during the navigation path capture and for AD restructuring.

The Exploration module comprises DocViewer, Exploration Recorder, Exploration Analyzer and Exploration Visualizer sub-modules. The DocViewer displays the pages of an AD and dynamically constructs the rating (R), tagging (T), commenting (C), expanding and collapsing features of the pages opened for viewing. The on-demand insertion of these features enables a clean separation of these features from the content of the pages. The Exploration Recorder is responsible for capturing and saving exploration paths and metadata. The metadata provides contextual information for an exploration, enabling it to be searched. The Exploration Recorder allows saved exploration data to be retrieved and displayed. The Exploration Analyzer supports analysis (encoding and aggregation) of the raw exploration data. The Exploration Visualizer provides functionality to visualize analyzed exploration paths as tree-views. It also provides search functionality to find saved exploration paths.

Fig. 2 (A) shows the main user interface of KaitoroCap, a plugin in the Confluence Enterprise Wiki. To start capturing an exploration path the user chooses the 'Start Exploration' menu item under the 'Tools' menu (Fig. 2 (A)) and fills in metadata (path name, keywords, role, reason of navigation, in relation to task) of the path (Fig. 2 (B)). The values chosen for the metadata should be relevant to the information-searching task as they serve as contextual information for an exploration and are used in the 'Search' feature to find suitable exploration paths.

By clicking on the 'Save' button, the metadata is saved and an exploration session started. In addition, the 'Start Exploration' menu item is changed to 'Stop Exploration' which can be clicked anytime to terminate the exploration session. Within an exploration session, the user can navigate to any page of an AD. KaitoroCap automatically inserts RTC, expand (read more)/collapse features into each section of the opened pages (for examples, Fig. 2 (C, D, E)). The sequence of navigation of the pages together with all the user interactions with the elements (RTC, expand/collapsing features and hyperlinks) on the pages is captured in the background without any interruption to the user's exploration of the AD. All this data is saved whenever the user navigates away from a page.

The exploration path can be retrieved as raw exploration data (Fig. 3 (A)), analyzed exploration data (Fig. 3 (B)) or as a tree-view (Fig. 3 (C, D)) by selecting the 'Retrieve Exploration Paths', 'Analyze Exploration Paths' and 'Exploration Paths Tree View' menu item respectively, from the main user interface of the tool (Fig. 2 (A)). The hierarchical tree-view can be collapsed ((Fig. 3 (C)) and expanded (Fig. 3 (D)) to provide both succinct and detailed views of the paths.

It can also be toggled to reverse between the two views. In addition, details of event data and the content of the visited sections in the tree-view can also be shown or hidden. Embedding the content of the visited sections in the tree-view visualization of an exploration path (Fig. 3 (D)) effectively makes the path a restructuring of the AD. Exploration path searching is supported by clicking on the 'Search Path' menu item from the main user interface of the prototype (Fig. 2 (A)). The user can search by providing search terms and choosing which metadata to search (Fig. 3 (E)). The tree-views of the resulting paths can be displayed side-by-side for comparison.

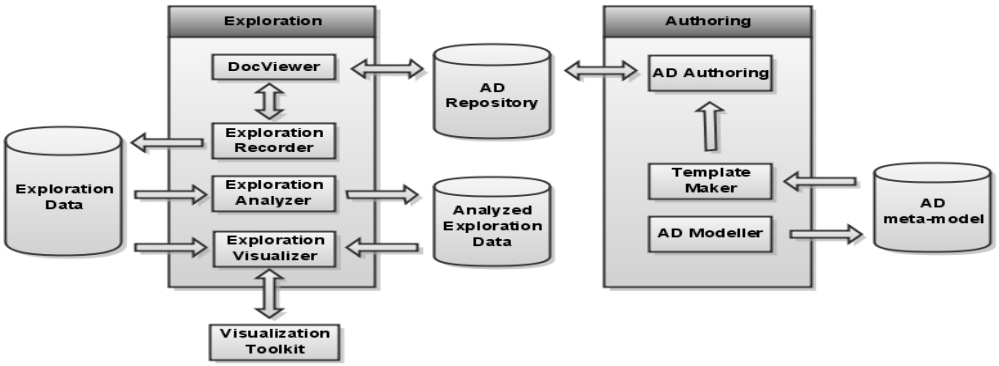

Figure 1. High-level design of KaitoroCap

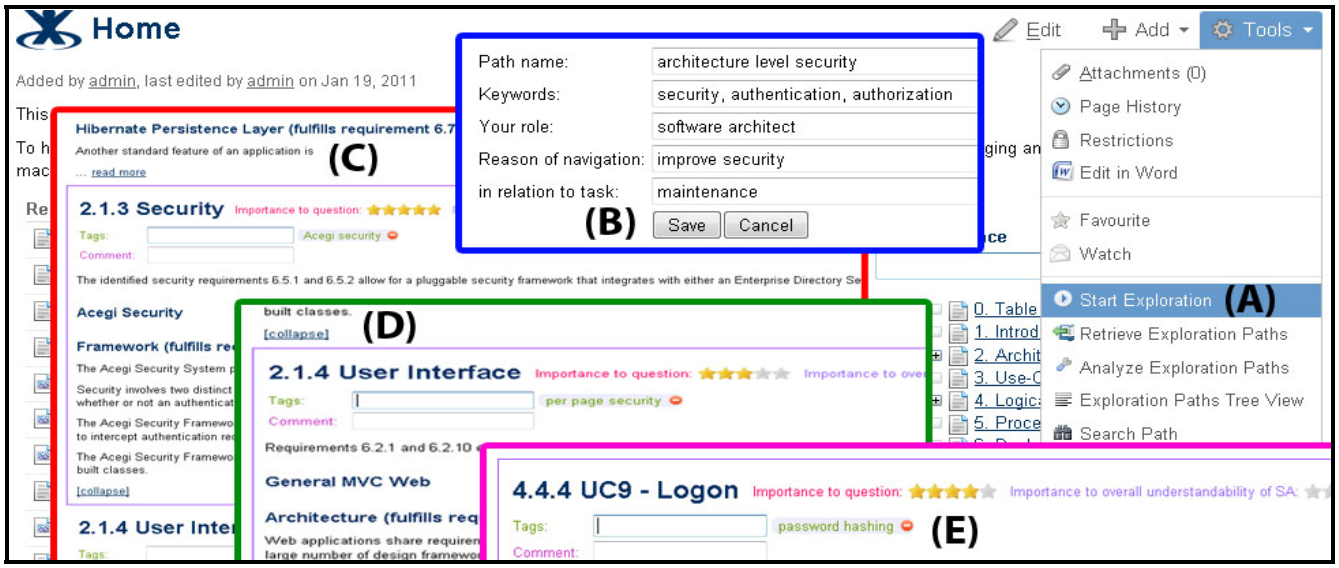

Figure 2. Main user interface of KaitoroCap (A); Metadata (B); Example of exploration (C, D, E)

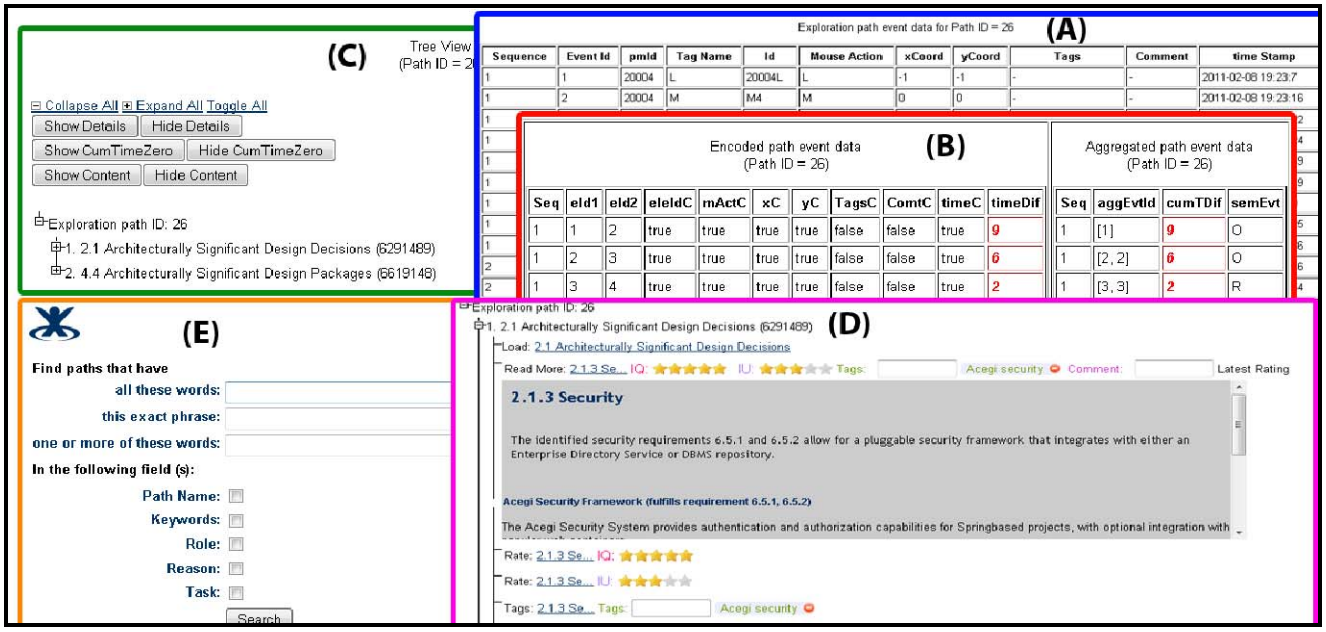

Figure 3. Raw (A) and analyzed exploration data (B); Collapsed tree-view (C); Expanded tree-view with embedded content (D); Search feature (E)

IV. IMPLEMENTATION

KaitoroCap was developed using several technologies including Atlassian plugin SDK, XWork Action for the controller, Java beans for the model, Velocity for the view, and JQuery for client-side scripting. The AD Authoring and DocViewer sub-modules are built on top of Confluence's functionality for creating and editing, and viewing of pages respectively. The former extends functionality to cater for the automatic creation of a page model for pages created or edited. The relevant Confluence XWork actions are overridden and chained to a custom action to generate the page model. A page model contains information that uniquely identifies a page, the sections and hyperlinks on the page. It also comprises the details of these items (for example, the page title, the sections' titles and contents, hyperlinks' texts and urls, etc). The DocViewer sub-module extends the existing page viewing capability via JQuery script to dynamically insert the rating, tagging, commenting, expanding and collapsing features into each section of a page opened for viewing. A section starts with a level 2 html heading and comprises this heading and all the succeeding elements before the next level 2 html heading. Each section is surrounded by a dynamic border, which is shown when the mouse pointer enters the section and hidden otherwise. The border provides visual cues regarding the section currently in focus for rating, tagging, commenting, expanding or

collapsing. The Exploration Recorder sub-module enables the user to capture exploration paths and save them together with contextual metadata (path name, keywords, role, reason, task). Other metadata such as start time, end time and path id is generated by the system and saved together with the paths.

During an AD exploration session, event data are generated from the interaction of the user with the elements within and across the pages. An element in this context refers either to a star rating, tag, comment, expand/collapsing feature or a hyperlink. Consequently, the types of interactions captured include clicking on the star rating features to rate a section, entering tags and comments, clicking to expand or collapse a section as well as clicking on hyperlinks to navigate to other pages. The series of event data items generated from the user interactions constitutes the exploration path data. A single event data item contains an event identifier (id), id and type of the affected element, information about the interaction (such as the type of interaction, timestamp and so on) and details of the page (the sequence of page within the exploration session and page model identifier). The Exploration Recorder makes use of JQuery to dynamically bind event handlers to the respective events (mouse click, mouse over, 'enter' key press) of the elements on the pages of AD. All the event data are later passed via AJAX to an XWork action class to be parsed and saved to the database. The saved exploration path data is displayed by populating a Velocity template.

In the Exploration Analyzer sub-module, exploration data is retrieved and analyzed in an XWork Action class. This involves encoding and aggregation of the raw exploration data and the determination of the semantic events. Encoding and aggregation enables the abstraction of the low-level interaction events in the event data into higher-level semantic navigation events. Encoding is performed by comparing each event data item with its succeeding event data item in the series. This is used to determine the occurrence of changes (in terms of element id, type of interaction, values, time stamp, etc) from one event item to the next. The encoded data is then analyzed for the possibility of event aggregation. Event data in sequence, same element id and type of interaction are candidates for aggregation.

The Exploration Visualizer sub-module displays the exploration paths in the form of hierarchical tree-views that are collapsible and expandable. The minutiae of an exploration path such as the details of the visited pages (for e.g. page title), the details of the visited sections (section title and content) and hyperlinks (text and url) are extracted from the respective page models and embedded in the tree-view. The Exploration Visualizer parses exploration data to build internal tree structures which are fed to the graphical display of the tree-views created using JQuery. It also provides the function to search for the exploration paths based on the metadata captured at the start of each exploration.

V. Conclusion and Future Work

KaitoroCap provides automated support to capture and use the AD exploration paths of others. This allows for reuse of knowledge across a team by reusing navigation paths through documentation found useful by others. By extracting the content of the visited sections from the page models and embedding the content inside the tree-view visualization of an exploration path, the path resembles a restructuring of the AD. A preliminary user evaluation of KaitoroCap has shown promising results in terms of capturing user's exploration paths, the tree-view visualization and searching of exploration paths. We will refine the prototype and build up a more substantial database of navigation paths for analysis using machine learning approaches. Different ADs will be used for exploration and another user study conducted to validate the usefulness of discovered chunks.

Acknowledgment

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

References

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

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] R. DeLine, M. Czerwinski, and G. Robertson, "Easing program comprehension by sharing navigation data", Proc. 2005 IEEE Symposium VL/HCC'05, 2005, pp. 241-248.

[9] 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.

[10] 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.

[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] 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.

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