A Tool to Visualize Architectural Design Decisions

Larix Lee and Philippe Kruchten

University of British Columbia
{llee,pbk}@ece.ubc.ca

Abstract. The software architecture community is shifting its attention to architectural design decisions as a key element of architectural knowledge. Although there has been much work dealing with the representation of design decisions as formal structures within architecture, there still remains a need to investigate the exploratory nature of the design decisions themselves. We present in this paper a tool that should help improve the quality of software architecture by enabling design decision exploration and analysis through decision visualization. Unlike many other design decision tools which acquire, list, and perform queries on decisions, our tool provides visualization components to help with decision exploration and analysis. Our tool has four main aspects: 1) the decision and relationship lists; 2) decision structure visualization view; 3) decision chronology view; and 4) decision impact view. Together, these four aspects provide an effective and powerful means for decision exploration and analysis.

1 Introduction

Software design is derived from making many decisions; capturing the most significant of these decisions would help convey significant insight and rationale behind the different aspects or features of the system architecture and design. However, the architectural knowledge provided by a simple enumeration of design decisions is often dry and difficult to peruse. If decisions can be browsed or visualized in an effective manner, the amount of time spent on communicating the software design with others can be reduced.

Defining software architecture to be a set of important design decisions [16] suggests that we need to effectively capture, browse, and exploit such design decisions. We addressed improving decision capture in an earlier paper [22], but for decision perusal and analysis, we propose that visualizing design decisions using different perspectives may improve the quality of decision information conveyed to the software architect, designer, or developer, which results in making better system architecture.

We have built a tool that assists the architects in decision exploration and analysis by employing different aspects to visualize a set of design decisions. This paper describes our tool and its various components as well as discusses how the tool contributes to architectural design decision exploration.

The paper is structured as follows: Section 2 provides some background into the scope of our research. Section 3 describes the previous work in visualizing and exploring software architectural design decisions, while section 4 describes our tool in detail. Sections 5 and 6 describe our experiences and our future work with the tool.

2 Background

During the software design process, we make many kinds of decisions. Some decisions can be traced directly to some element in the design or in the code. Other decisions are more difficult to track down to any concrete artifact as they span over many different elements or specify certain properties of the design/system. There are also decisions related to business/organizational issues. People frequently change their minds on decisions and the shift to agile software development methods imply that decision changes will be more common in occurrence. A single decision change could significantly affect the entire architecture of the software system being developed, implying the need to clearly document and model the role of design decisions and knowledge into software architecture.

2.1 Design Decisions and Software Architecture

What differentiates architectural design decisions from other design decisions is that the former are decisions that cross-cut multiple components and connectors and intertwine with other design decisions [14]. If one of the architectural decisions is changed, then the components dependent upon the changed decision would be impacted; furthermore, the change may affect other decisions that are intertwined with or related to the changed decision. Representing decisions as explicit, formal decision entities within architecture and their descriptions may assist in determining the severity and scope of a change. Decision entities make interdependencies more apparent and helps identify the set of decisions and architectural components that cause design rule or constraint violations.

The current shift is towards making design decisions and assumptions explicit within architectural descriptions [11, 14, 17]. Since these decisions intertwine and cross-cut architectural components, a decision view into software architecture [11] is fitting and we can model software architecture involving the use of formal decision entities. Software architectural models that use explicit decisions or assumptions include Lago and van Vliet's assumptions meta-model [19], the Archium metamodel [14], the ADDSS metamodel [9], and the architectural decision ontology described by Ackerman and Tyree [2].

2.2 Design Decision Representation

Many architectural models identified that the decision itself should have a model of its own to represent the architectural knowledge it carries. There are two main streams in the capture and representation of architectural knowledge as design decisions. The first stream is to use an argumentative approach via design rationale and the second stream is to use structured decision entities.

2.2.1 Design Rationale

Design rationale documents the analysis history of why particular artifacts or features are chosen [21]. They can also refer to non-functional requirements and constraints imposed on the general nature of the system. Design rationale commonly employs the use of an argumentation structure, which improves the capturing process as the knowledge can be expressed in familiar forms such as issues, alternatives, questions, and options [12]. In essence, they take the perspective that decisions are context dependent, so the context and background must be captured in detail. Although the decisions can be captured using comfortable and relatable structures, it is difficult to extract the architectural decisions from the rationale as the decisions are embedded within the justification texts of the design rationale. The more expressive form of using design rationale makes decision referencing more difficult.

2.2.2 Explicit Decisions

The second stream is representing design decisions explicitly, viewing the choice as primary and the context and justification secondary to that decision. Although research in the design rationale community has dealt with representing decisions and assumptions explicitly, such as SIBYL [20], the software architecture community developed this area significantly due to the software architectural shift towards making design decisions and assumptions explicit. We introduced an approach that makes decisions first-class citizens by representing decisions using a decision ontology model [17]. Jansen and Bosch's architectural model [14] as well as Lago and van Vliet's work [19] also makes contributions to what is considered part of the decision model.

Though there are multiple approaches in representing design decisions and other architectural knowledge, the definitions are starting to converge. An attempt to define relevant architectural knowledge [8] by generalizing the principles of what would constitute architectural knowledge brings together other definitions of architectural knowledge, such as our decision ontology model [17], Bosch's four decision aspects [5], and Tyree and Ackerman's decision description template [24]. A recent literature survey by de Boer and Farenhorst [10] collected and synthesized definitions of architectural knowledge to conclude that all authors considered design decisions to be a significant part of the knowledge, and more definitions will come as the concept matures.

3 Decision Exploration and Visualization Tools

The formalized structure of explicit design decision representation in software architecture offers high decision analysis and exploitation potential. However, the analytical and exploitative capabilities of architectural design decision representation are bound by the way the information is organized or rendered. If the decisions are not effectively conveyed to the software architects, designers, and developers, then we question the usefulness of capturing those decisions; consequently, architectural design decision tools implementing explicit decision representation would need to adequately support decision exploration.

Design decision visualization facilitates understanding of the architecture and allows a kind of "walkthrough" of the designers' intents. Visualization is one of the five requirements listed in the decision view of software architecture [11]. These requirements are: multi-perspective support, visual representation, complexity control (in terms of scalability and navigation), groupware support, and gradual formalization of design decisions. The five requirements were described later to apply to all tools that utilize or manage design decisions [9].

3.1 Current Design Decision Tool Support

The recent interest in design decisions stimulated the development of several decision-based architectural tools. There are a number of tools created recently for the exploration and analysis of design decisions; some are from the design rationale community, some are from the architecture community. We will briefly look at a few of these tools in the context of decision visualization and exploration.

Although it is a design rationale tool, the SEURAT tool is an Eclipse development environment plug-in utility that captures and utilizes design rationale by linking its software code [7]. Decisions are visualized as part of the rationale in hierarchical tables displayed in Eclipse "views". Since the goal of SEURAT is to assist in software maintenance, the application to software architecture is not explicitly made. Another rationale-based tool, Sysiphus, is a toolset that assists in the capture of various system models for system development activities [6]. It supports rationale-based design decisions and links them with system models. However, with the focus being on collaboration support and on multiple system models, decision relationships and states are not investigated in detail.

The Archium tool is an architectural design decision tool which primarily focuses on how design decisions can be traced to the requirements and to the architectural components of a software architecture [15]. Although the tool's decision visualization component uses a graphical view of decision relationships, Archium regards design decisions as a "change function" with a single parameter [14], and the visualization of the decision entities themselves are light. Decisions are visualized with a dependency graph and the properties of the decision are listed in a table of attributes. Each decision can be linked to a graphical presentation of the architectural model, showing the components and connectors that relate to the design decision.

The ADDSS tool is a web-based tool to capture and document architectural design decisions for immediate browsing [9]. The tool lists the system requirements, the decisions and the requirements it addresses, and user-uploaded picture files representing the architectural products in a table format. Currently, the uploaded picture files from ADDSS approach graphical representation of architecture; however, another version is being developed to address the requirements stated in their follow-up paper [8]. Another web-based design decision tool, known as PAKME, focuses on general architectural knowledge capture and management of scenarios, patterns, design options, and decisions for the software architecture process [3, 4]. All these components, including decisions and their relationships, are displayed in tables and are retrieved by query-based mechanisms.

Although not officially designed to be tool-based work, there was a case study that focuses specifically on the ontological visualization of design decisions [18]. Design decisions followed the decision ontology model [17] and applied a visualization framework that visually clusters decision entities together depending on the query. The case study revealed that many architectural knowledge use cases can be supported by the clustering tool, but further tool extension is needed to explore the ideas of inter-decision relationships and the amount of information that relationship analysis can provide.

IBM research has also developed a tool, called the Architect's Workbench, which acknowledges the importance of decisions, but like many other tools, it has no obvious ways to visualize webs of decisions [1]. There are other successful knowledge visualization tools that document decisions such as the Compendium tool [23], which documents and visualizes the flow of knowledge and design rationale during interactive team meetings. However, these tools are of a general scope and do not apply well to the interrelated, dynamic nature of software architectural design decisions and the multiple perspectives these decisions can have.

In many of the tool examples above, decision visualization would enable the attention to be directed towards specific areas of interest where useful conclusions may be drawn; yet, the majority of the tools did not significantly investigate the concept of decision visualization as a separate decision representation view or component. Visualizing the decisions using different perspectives may improve the quality of the software design by helping software architects, designers, and developers understand the nature and impact the design decisions they made.

4 Tool Implementation

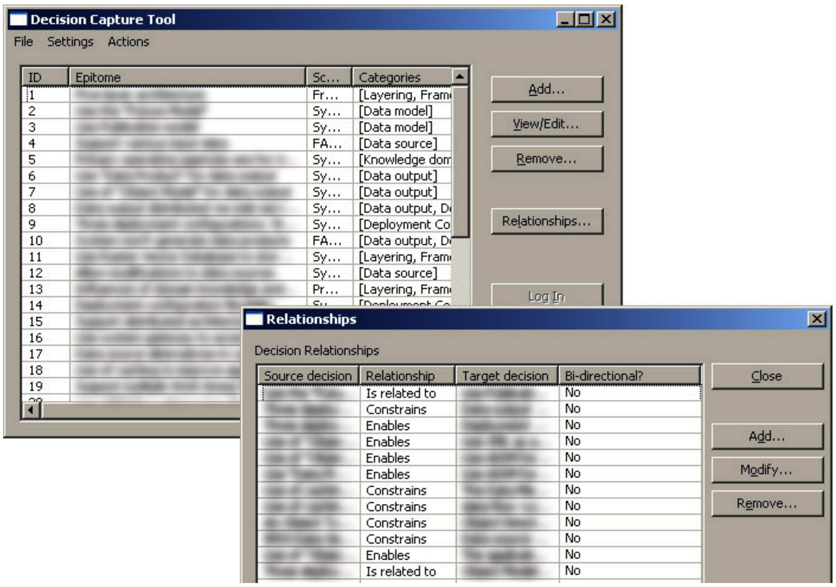
We implemented a tool that visualizes software architectural design decisions separately from the software architecture in which they are referenced. The purpose of this tool is to facilitate both decision browsing and detailed decision analysis.

Although there are various models to represent software architectural design decisions, we excluded rationale-based decision models since those models detract attention away from the core decisions. We are primarily interested in how architectural design decisions can be exploited, so a simpler, broader model that addresses software design decisions in general is preferred. Decision models that architects and designers can use more easily during software development are ideal. We decided to adopt Kruchten's decision model [17] because the model is simpler, more process-focused, and it is the only model that explicitly represents decision *relationships*.

The tool captures design decisions and stores them into a file or a database for later retrieval. Moreover, the decisions can be imported or exported using XML across multiple computer workstations to ease decision capture and distribution. The user can create, modify, remove both decisions and their interrelationships, and visualize the decisions in several ways to support decision perusal and analysis. The tool utilizes the Prefuse visualization framework [13] for the visual representation of design decisions. The tool has four main views for decision visualization and information display. The first is a simple tabular list of decisions and their relationships, while another view visualizes the decisions using decision-graphs to display the decision structures and relationships. The tool can also visualize the decisions in a chronological order. The fourth view displays decisions from an impact perspective.

4.1 Decision / Relationship List

This view, a *decision table*, is the most common in the decision tools, and almost every tool mentioned in Sect. 3 supports this view. The decision / relationship list simply lists the design decisions in a table, showing a selection or all the attributes of a design decision. Decision relationships are also listed in another table that references the decision list. A screenshot is depicted in Fig. 1. The purpose of this view is



ID	Epitome	Sc...	Categories
1	[redacted]	Fr...	[Layering, Fram
2	[redacted]	Sy...	[Data model]
3	[redacted]	Sy...	[Data model]
4	[redacted]	FA...	[Data source]
5	[redacted]	Sy...	[Knowledge dom
6	[redacted]	Sy...	[Data output]
7	[redacted]	Sy...	[Data output]
8	[redacted]	Sy...	[Data output, D
9	[redacted]	Sy...	[Deployment Co
10	[redacted]	FA...	[Data output, D
11	[redacted]	Sy...	[Layering, Fram
12	[redacted]	Sy...	[Data source]
13	[redacted]	Pr...	[Layering, Fram
14	[redacted]		
15	[redacted]		
16	[redacted]		
17	[redacted]		
18	[redacted]		
19	[redacted]		

Decision Relationships

Source decision	Relationship	Target decision	Bi-directional?
[redacted]	Is related to	[redacted]	No
[redacted]	Constrains	[redacted]	No
[redacted]	Enables	[redacted]	No
[redacted]	Enables	[redacted]	No
[redacted]	Enables	[redacted]	No
[redacted]	Enables	[redacted]	No
[redacted]	Constrains	[redacted]	No
[redacted]	Constrains	[redacted]	No
[redacted]	Constrains	[redacted]	No
[redacted]	Constrains	[redacted]	No
[redacted]	Enables	[redacted]	No
[redacted]	Is related to	[redacted]	No

Fig. 1. Decision and relationship lists showing the current set of design decisions and their relationships: the user can create, view, modify, or remove design decisions directly from the tool. In this figure, the "relationships…" button has been pressed, bringing up the relationships dialog that is currently displayed in the foreground. The decision epitome has been intentionally concealed to protect the intellectual property of the organization that provided the dataset.

to supply a quick and effective way to browse and retrieve information from design decisions. The textual representation of the decisions facilitates decision querying and simple decision entry. However, it is difficult to trace decision relationships and quickly assess decision properties when the decision set gets large.

4.2 Decision Structure Visualization

With large decision sets, an effective way to sort and analyze decision information is to represent the decisions graphically. In this view, we visualize *decision structure* as graphs, in which decisions are represented as nodes and the relationships are the edges. Figure 2 depicts a decision graph that represents the decisions and their relationships. Decisions and relationships can be created, selected, viewed, modified, and removed from this view. The advantages of graph visualization are apparent, such that an observer can see relationships and their associated decisions more quickly than from a list.

Besides the view's graphical visualization, there is a high degree of interactivity to communicate information. Using a force-directed layout for the visualization of the decision graph, the tool represents decisions of a less mature state as being physically lighter in the layout model and visually smaller than more mature decisions. Decision maturity refers to the decision states in Kruchten's decision ontology model, where a decision can be an idea, tentative, decided, approved, challenged,

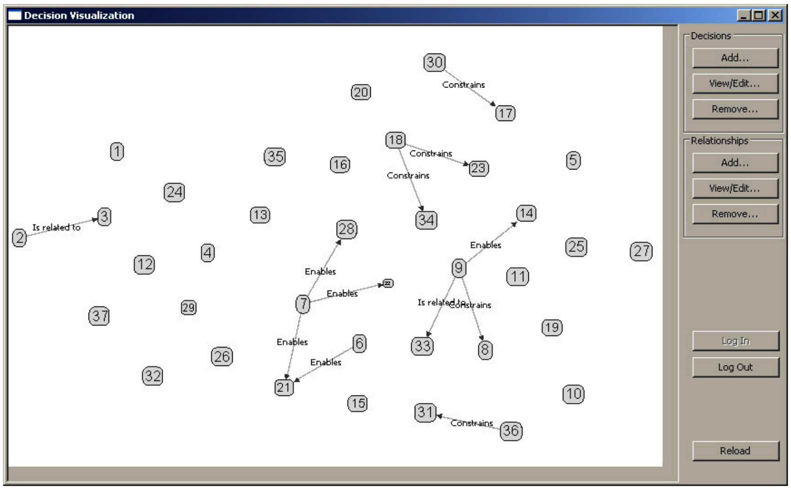


Fig. 2. The visualization of design decisions and their relationships as a directed graph: the nodes represent the decisions and the directed edges represent the decision relationship to another decision. The node-size depends on the state of the decisions. Not shown in this figure are the semantic zooming properties that can list decision properties like the decision's epitome, and the interactivity provided visualization where decisions at less mature decision states can be moved around the visualization more easily than other decisions.

rejected, or obsolete. Visually, the maturity of a design can be assessed from the number of small or large nodes in the graph. However, when the user interacts with a decision node or a cluster of nodes, the user can quickly assess the maturity from how quickly the decision can be moved around the screen. For example, more mature decisions are heavier and more difficult to move, so the decisions behave like heavy objects.

Depending on the zoom level, the decision nodes can show more or less information about the decision. When a user zooms towards a decision, the decision's properties will appear inside the node. When a user zooms away, decision information gets hidden. Viewing the decision or relationship details can also be performed without zooming simply by selecting a decision.

4.3 Decision Chronology Visualization

The tool supports a time-based view of design decisions, the *decision chronology*, to show the evolution of design decisions and provide the ability to quickly determine created or changed decisions during a specified time interval. This view is shown in Fig. 3. A user can select a subset of these decisions to view more closely, such as the decisions within a cluster, and can create, view, or modify decisions. This feature is especially valuable when there are periodic reviews of the architecture: it saves time and effort for the reviewers who are already familiar with the system, who may only want to know, "What has changed since last time?"

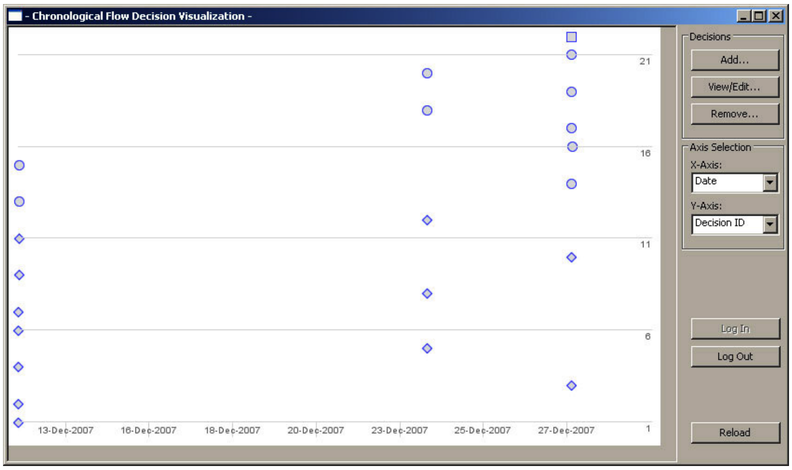


Fig. 3. The chronological view of a set of design decisions: this example shows three decision creation or activity sessions over a two-week interval. The state of the decisions is denoted by the shape: Diamonds are idea, circles are tentative; and squares are decided.

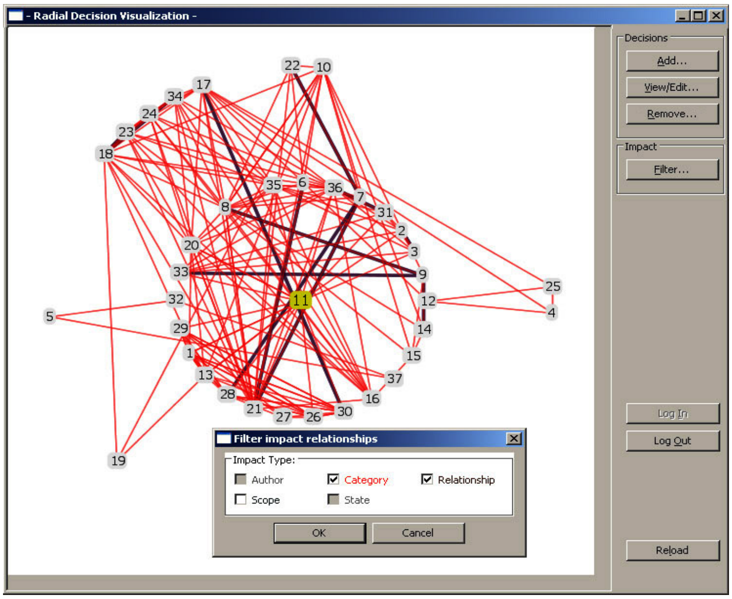
This view initially displays all the decisions created and modified during the project in a timeline, with the date on the x-axis and a user-selectable field for the y-axis. Decisions that are closely spaced denote a decision capture or activity session. A user can quickly identify the state of a decision by its shape in the view.

A particular area of interest is in the user-selectable y-axis. The tool currently allows categorization of the y-axis by decision ID or decision author. If the decision ID is used for the y-axis, one can view decision changes in a global perspective (as the decision ID is implemented as an increasing number). If the author is used for the y-axis, we can determine which decision-makers are most active and which changes they have made. Categorizing by author include the ability to find both subversive and critical stakeholders who can potentially damage the system if they change their minds [18]. With other category types for the user-selectable y-axis, the tool can be a powerful way to exploit hidden knowledge within design decisions.

4.4 Decision Impact Visualization

The fourth view of design decisions that this tool supports is the *decision impact*. Shown in Fig. 4, this view provides a visualization of decisions that can be potentially impacted by a change of a decision. This view is very valuable when radical changes are about to be made to a system, and the impact of certain changes may not be obvious in the architectural design or the code. Although the decision structure visualization supports visualization of decision relationships, there are related decisions that are associated by attributes, such as author, scope, and categories. The tool provides an entry-point into the large matrix of potential impact-relationships by visualizing it.

The decisions are laid out using a radial layout, where all other decisions surround the selected center decision. Selecting a different decision brings that decision into the center and all other decisions surround it. Resting a mouse cursor on a decision would



Filter impact relationships

Impact Type:

Author [✓ Category] [✓ Relationship]

Scope State

Fig. 4. The decision impact view of design decisions: the nodes represent design decisions while the colored lines represent the impact-relationships between them. Thick edges are the decision relationships in our decision ontology model [17], thin edges are impact-relationships.

highlight neighboring decisions associated with an impact-relationship. The impact relationships can be filtered according to different criteria, such as category, scope, or relationship. Currently, the tool links decisions that share a common criteria value with an impact-relationship, though the tool can be modified to support different criteria values, ranges, and thresholds.

5 Experience with the Tool

We were able to use industry datasets to test the practicality of the decision visualization aspects. We demonstrated and used the tool with industry participants and obtained feedback on the feasibility of the visualization tool.

One group of industry participants is from a large technology corporation that is both process and documentation heavy. The participants are involved in a multi-national project to develop an elaborate spatial modeling system, but the system is constrained by many domain-specific guidelines and procedures. As the project is very large, we reduced the scope and focused the decision capturing on the deployment configurations and the data model used for this system.

After demonstrating the tool to the participants using their own decision dataset, we asked the participants what their initial impressions are. They found that the decision and relationship lists were acceptable, but for the graphical decision structure visualization, the participants stated that the decision identifier used in the default zoom-level is not fully intuitive, as it can be hard to mentally map decisions details to the decision identifiers. The participants mentioned that the decision-relationship graphs are informative, but they reported that the explicit decision relationships are difficult to elicit and categorize, partly due to the various relationship definitions and the tacit nature of defining these relationships. The participants appreciate the ability to see decision sessions in the decision chronology view, and they found the “author” criterion for the user-selectable y-axis to be an interesting application. Although the participants felt that implementing a fine-grained filtering mechanism would improve usability, all the participants agree that the decision impact view can be effective in identifying potential, indirectly-impacted decisions.

From the initial feedback, we were able to validate the feasibility of the visualization tool with industry datasets in a laboratory setting. However, we would like to validate the tool in industry, allowing software designers, architects, and developers to use the tool firsthand and investigate the exploratory and analytical capabilities of the visualization tool.

6 Future Work

The four aspects that the visualization tool supports provide the user with a powerful means to explore and analyze decisions with. This user can be a software architect, a designer, a developer, or any other individual who would like to learn and explore the decisions for a project. We are continuing to develop the tool further, improving the user interface and providing more useful features to support the exploration and analysis of decisions. Full query-support is a feature we would like to implement. Another useful feature is to link decisions across the four different views. Selecting a decision in one view should select the same decision in another view. This way, the user can maintain continuity between views and may discover new associations about the design decisions. We would also like to implement fine-grained impact querying as suggested by the participants, after which we could implement multi-level impact visualization to improve usability.

Although we have identified four aspects, there may be other visualization techniques that can reveal more information provided by the set of decisions. Ontological visualizations [18] can be added to assist in decision retrieval and categorization, while support for visualizing software and organizational artifacts may provide insight by associating decision capture with the software development process. Furthermore, visualizing the links between artifacts and design decisions may improve traceability between requirements, architecture, and developed software code.

7 Conclusion

The tool we presented here has been developed and used with industry datasets from several software organizations. The screenshots displayed in the preceding figures demonstrated early results on the feasibility of the visualization tool as applied to industry, but further evaluation is needed to identify additional use cases in which the visualization tool may be useful for industry practice. The decision sets acquired from industry show some encouraging early results, but further detailed case studies with industry participants may be necessary to apply other visualization techniques to better capture, represent, and relay software architectural knowledge.

References

1. Abrams, S., et al.: Architectural thinking and modeling with the Architects' Workbench. IBM Systems Journal 45(3), 481–500 (2006)
2. Akerman, A., Tyree, J.: Using ontology to support development of software architectures. IBM Systems Journal 45(4), 813–825 (2006)
3. Babar, M.A., Gorton, I., Jeffery, R.: Capturing and Using Software Architecture Knowledge for Architecture-based Software Development. In: 5th International Conference on Quality Software (QSIC), Melbourne (2005)
4. Babar, M.A., Gorton, I., Kitchenham, B.: A framework for supporting architecture knowledge. In: Dutoit, A.H., et al. (eds.) Rationale Management in Software Engineering, pp. 237–254. Springer, Heidelberg (2006)
5. Bosch, J.: Software Architecture: The Next Step. In: Oquendo, F., Warboys, B.C., Morrison, R. (eds.) EWSA 2004. LNCS, vol. 3047, Springer, Heidelberg (2004)
6. Bruegge, B., Dutoit, A.H., Wolf, T.: Sysiphus: Enabling Informal Collaboration in Global Software Development. In: First International Conference on Global Software Engineering, Costao do Santinho, Florianopolis, Brazil (2006)
7. Burge, J.E., Brown, D.C.: Rationale-based Support for Software Maintenance. In: Dutoit, A.H., et al. (eds.) Rationale Management in Software Engineering, pp. 273–296. Springer, Heidelberg (2006)
8. Capilla, R., Nava, F., Duenas, J.C.: Modeling and Documenting the Evolution of Architectural Design Decisions. In: Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, IEEE Computer Society, Los Alamitos (2007)
9. Capilla, R., et al.: A web-based tool for managing architectural design decisions. SIGSOFT Software Engineering Notes 31(5) (2006)
10. de Boer, R.C., Farenhorst, R.: In Search of 'Architectural Knowledge'. In: Third Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, IEEE Computer Society, Germany (2008)
11. Duenas, J.C., Capilla, R.: The Decision View of Software Architecture. In: 2nd European Workshop on Software Architecture, Morison, Italy (2005)
12. Dutoit, A.H., et al.: Rationale Management in Software Engineering: Concepts and Techniques. In: Dutoit, A.H., et al. (eds.) Rationale Management in Software Engineering, pp. 1–48. Springer, Heidelberg (2006)
13. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: SIGCHI conference on Human factors in computing systems (2005)
14. Jansen, A., Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: Working IEEE/IFIP Conference on Software Architecture (WICSA) (2005)
15. Jansen, A., et al.: Tool support for architectural decisions. In: Working IEEE/IFIP Conference on Software Architecture (WICSA 2007), Mumbai (2007)
16. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley, Boston (2003)
17. Kruchten, P.: An Ontology of Architectural Design Decisions. In: 2nd Groningen Workshop on Software Variability Management, Rijksuniversiteit Groningen, NL (2004)
18. Kruchten, P., Lago, P., van Vliet, H.: Building up and reasoning about architectural knowledge. In: Hofmeister, C., Crnković, I., Reussner, R. (eds.) QoSA 2006. LNCS, vol. 4214, pp. 43–58. Springer, Heidelberg (2006)
19. Lago, P., van Vliet, H.: Explicit Assumptions Enrich Architectural Models. In: International Conference on Software Engineering (ICSE 2005), ACM Press, USA (2005)

20. Lee, J.: SIBYL: a tool for managing group design rationale. In: ACM conference on Computer-supported cooperative work (CSC1990), Los Angeles (1990)
21. Lee, J., Lai, K.-Y.: What's in Design Rationale? In: Design Rationale: Concepts, Techniques, and Use, pp. 21–51. Lawrence Erlbaum Associates, Inc, Mahwah (1996)
22. Lee, L., Kruchten, P.: Customizing the Capture of Software Architectural Design Decisions. In: 21st Canadian Conference on Electrical and Computer Engineering, IEEE, Los Alamitos (2008)
23. Selvin, A., et al.: Compendium: Making Meetings into Knowledge Events. In: Knowledge Technologies, Austin, TX (2001)
24. Tyree, J., Ackerman, A.: Architecture Decisions: Demystifying Architecture. IEEE Software 22(2), 19–27 (2005)