Software Architecture Rationale Capture through Intelligent Argumentation

Xiaoqing (Frank) Liu, NagaPrashanth Chanda and Eric Christopher Barnes
Computer Science Department
Missouri University of Science and Technology
Rolla, MO, USA
{fliu, nc9r6, ecbyt7}@mst.edu

Abstract—A growing model for software architecture defines it as a set of principal design decisions which describe the system. These design decisions need to be made by resolving design issues in a collaborative environment that helps software architects to design the architecture of a system. The architecture design decisions are usually made based on experiences since there aren’t defined methods and models for architecture design. Each design decision yields a set of outcomes which impacts both the system architecture and the final product. As software product systems tend to be large in size, one needs to understand the rationale behind decision of each architectural element. This is to justify the system’s design and to avoid critical architectural problems. Often during these design decision making process the rationale is not fully captured. This paper identifies and addresses the above mentioned research challenge. It presents a method for software stakeholders to use intelligent argumentation system for collaborative rationale capture. The argumentation will be recorded in an online system to document the rationale behind the design decisions resulting in product architecture. Finally, the proposed method is evaluated using a case study. It demonstrates feasibility of capturing software architecture rationale using intelligent on-line argumentation.

Index Terms— Collaborative Software Architecture Design, Collaborative Decision Support, Intelligent Argumentation, Collaborative Knowledge Management, Web-Based Knowledge Management

I. INTRODUCTION

During the design of the architecture for a system, there are many design issues that need to be addressed. The stakeholders of an organization have their own rationale from various perspectives to resolve design issues. Most of the time their views to solve a design issue conflict with each other. In order to collaboratively examine and resolve an issue, stakeholders must participate in argumentation to exchange their rational with respect to their design choice.

Argumentation is a process by which stakeholders choose from multiple alternatives to resolve a design issue, giving arguments explaining their support or lack of support for a given design alternative. This provides stakeholders with an equal platform for exchanging their knowledge and viewpoints, allowing them to select an alternative which best address the concerns of the whole. If properly organized and recorded, argumentation provides a format for rationale which gives a comprehensive explanation of the motivations behind design decisions.

We developed an argumentation system that allows stakeholders to post their arguments in support or attack of an alternative that addresses the issue. In order to strengthen the arguments, stakeholders can also post evidences along with their arguments. During this process, as the number of arguments increase, a huge argumentation tree is produced. To compute the design alternative that is favored by most of the stakeholders, the entire argumentation tree is analyzed using argumentation reduction inference engine. The computed result helps the decision makers to select an appropriate alternative to resolve the architecture design issue.

As the architecture of the system evolves, there are many architectural elements and their respective design decisions. So, there must be proper mapping between the elements of the architecture and design decisions. This allows the users to trace from architectural elements to their associated design issues, design alternatives, arguments, evidences, and decisions in an argumentation tree.

This paper is organized as follows. Section II presents a review about the related work, Section III briefs about the intelligent argumentation system that is used for argumentation process, Section IV explains in detail about the framework to capture software architecture rationale using intelligent argumentation and Section V validates the proposed method using a case study.

II. RELATED WORK

This section presents the literature review of related research. Sub section A presents the work done in field of architectural knowledge management and Sub section B discusses the related work of other argumentation systems.

A. Architecture Knowledge Management:

Much work is done in field of architectural knowledge management. In one of the approach proposed by Fabian et al [2], design decisions are concrete bindings between requirements and from requirements to their manifestations as model elements in architectural models. The architectural knowledge rationale is maintained as documentation linked to architectural significant requirements. Another approach PAKME [3] captures design alternatives as cases from literature. A design case consists of problem and solution, patterns and tactics used, rationale, and related design options. ADDSS [4] is another system focused on storing the rationale behind design decisions. It captures rationale by linking
motivating factor to design decision made, also stores design pattern knowledge.

Cui et al [5] proposed a design centric architectural design in which stakeholders determine the architectural issues from requirements as well as their solutions. The system explores all feasible combinations of the issue solutions and combines the feasible combinations to generate architectural solution. The rationale is automatically collected from each solution involved in the final architecture. While the rationale includes both pros and cons of a particular solution, it is not presented in great detail. Architecture as design decisions was presented by Lytra et al [6]. The system automates component and constraint generation based on design decisions. Each design decision has a set of outcomes, which are mapped directly to the architectural elements they generate in the component model.

Savolainen and Männistö [7] proposed a method of creating architectural views that more prominently communicate the conflicts from multiple perspectives between key stakeholders' concerns. It helps capturing architecture rationale based on interactions among stakeholders. Bratthall et al [8] experiment to verify the importance of design rationale when predicting change impact is mostly related to how to document the architectural rationale and how important is the availability of architectural knowledge to assess the change impact.

B. Argumentation Systems

There is large amount of work done in collaborative decision making through argumentation. Most of the argumentation systems proposed follow Stephen Toulmin’s model of argumentation [9]. The first method was gIBIS [10] which represents design dialog as graph. The method displays arguments, issues and positions in form of graph. HERMES [11] is a computer based decision support tool which organizes arguments and evidences in hierarchy. Chenn-Junn Huang’s argumentation system assesses the quality of the arguments by parsing the arguments [12].

The above methods related to architectural knowledge management are not widely adopted in practice. The knowledge management methods discussed above captures only static knowledge of the architecture, not the dynamic exchange of rationale of stakeholders on design issues. The methods fail to capture the knowledge from multiple perspectives. The methods related to argumentation systems are not specific to architectural knowledge management and are difficult to use.

III. THE INTELLIGENT ARGUMENTATION SYSTEM

We developed Web based Intelligent Argumentation system for collaborative decision support [1]. The stakeholders can participate in the argumentation process using web browser and the server manages different clients simultaneously along with business calculations to determine the stakeholders’ favorability for a design issue. Within the argumentation system, the design issue is referred to as an Issue and various design alternatives that addresses the issues are called Positions. The stakeholders can post their views in terms of arguments during the argumentation process. The stakeholders should enter a degree of strength explicitly for an argument in range of -1 to 1. An argument with negative degree signifies that it is attacking another argument and argument with positive degree signifies that it is supporting another argument. An argument with 0 degree strength signifies indecisiveness [13]. The priority of the stakeholder is also one of the factors in decision making. So, the stakeholders are assigned priorities based on their role during software design process [15]. Figure 1 shows a snapshot of the intelligent argumentation system.

![Figure 1 Snapshot of The Intelligent Argumentation System](image)

The entire argumentation tree built during the argumentation process is reduced to a single level such that all arguments posted refer directly to the position. The overall favorability of the design alternative is computed by weighted summation of the argument strengths. In order to assess the impact of the indirect arguments on a position, we have four general argumentation reduction heuristic rules and 25 fuzzy rules are derived from these 4 rules. Please refer to references [1, 14] for more discussion of the intelligent argumentation system. The four heuristic rules are formulated as,

**Argumentation Reduction Rule 1:** If argument B supports argument A and argument A supports position P, then argument B supports position P.

**Argumentation Reduction Rule 2:** If argument B attacks argument A and argument A supports position P, then argument B attacks position P.

**Argumentation Reduction Rule 3:** If argument B supports argument A and argument A attacks position P, then argument B attacks position P.

**Argumentation Reduction Rule 4:** If argument attacks argument A and argument A attacks position P, then argument B supports position P.

Figure 2 shows a sample argumentation tree which represents positions, arguments and their associations. There are two design alternatives for a design issue, Position 1 and Position 2. Arguments A1, A2, A3, and A4 are posted under position 1. The argument A3 attacks position 1 based on an argumentation heuristic rule.
IV. FRAMEWORK TO CAPTURE SOFTWARE ARCHITECTURE RATIONALE THROUGH ARGUMENTATION

The proposed method captures the software architecture decision rationale through intelligent argumentation. Unlike other methods discussed earlier, where they capture static knowledge of the architecture, our method is dialog based which captures dynamic interactions of the stakeholders who are responsible for the architecture design. It is built on an existing intelligent argumentation system which has its applications in various domains and displayed promising results.

The requirements for a software product are the basis for the development of a system. The product needs to be developed such that it satisfies the requirements of its customers. The software architect design team has to design the architecture of the system before implementation can begin. The architecture design is mainly dependent on the software requirements captured during requirements gathering stage. The problems that the system faces to ensure the satisfaction of the requirements are termed as design issues. These design issues need to be resolved through one or more design decisions. Figure 3 shows the architecture of the present method to capture software architecture rationale knowledge through intelligent argumentation.

The architecture of the proposed method consists of three high level components. The first is the issue based architecture design rationale capture, which is responsible for documenting the rationale behind the design decisions in terms of arguments from various stakeholders for different design alternatives. The argumentation based software architecture design rationale knowledge base shows the linkage between the software architecture, its design issues and their respective rationales. Lastly the inference engine determines design decision by performing inference process on the argumentation tree.

The internal structure of one of the components, the argumentation based software architecture knowledge base is shown in Figure 4. The structure includes software architectural elements and rationale elements such as design issues, design decision, arguments, evidences and design alternatives. It clearly shows the dynamic interactions between the stakeholders to determine the design decision. There exists a complex relationship among various elements. Some of such relationships include: the design issue has its origin from requirements and architectural elements, the requirements determine the architecture of the system, arguments are posted for a design issue and design alternatives, evidences are attached to arguments, design alternatives solve a design issue and design decision affects the architecture of the system.

Here, the software architecture is considered to be designed using four elements namely components, connectors, configuration and constraints. A component is like a service that provides some required functionality and is derived from functional requirements. A connector is a link between two components of the system to depict the interactions among them. For example: a broadcast connector that can transmit the required information between two components. Constraints determine the qualities that the components or connectors...
should satisfy. They are generally attached along with the architectural elements that need to satisfy particular quality. Configuration is a collection of components and connectors.

As stated earlier, design issues are derived directly from software requirements. The design decision that addresses the design issue may have its impact on multiple architectural elements. Therefore, once the design decision is taken, the architecture of the system needs to be modified accordingly. There are potentially multiple design alternatives to solve a given design issue. A design alternative consists of a design solution along with the set of architectural elements it affects. Deciding which alternative should be implemented may be a contentious issue. If the conflict among the stakeholders is properly managed, the issues can be resolved to achieve a design decision that best addresses the concern of the whole group and also will have a structured rationale behind the decision. Once multiple design alternatives are suggested the stakeholders debate the pros and cons of each in terms of their ability to satisfy the requirements.

After entire argumentation tree is built, the inference engine determines which design alternative that is favored by most of the stakeholders. This favorability factor helps the stakeholders to decide which design alternative to implement in the architecture design. Each architectural element such as component, connector and constraint maintains a link with the design decision which resulted in their origin and also to the design decision which might have modified afterwards. The captured rationale enables the stakeholders to understand the architecture of the system from requirements to its design.

During software architecture design, the functional requirements define what the system should do and they contribute mainly to the design decisions to be taken. The non-functional requirements (NFR’s) are equally important as they define how the system should be in order to perform desired functionality. Each architectural element in system’s architecture contributes in some way to the overall satisfaction of the systems NFRs [16]. Often, majority of the time is allocated to resolve design issues pertaining to functional requirements without considering NFR’s quoted. The quality of a software system is measured with respect to how well the system architecture is designed in order to satisfy the NFR’s. For instance, a system that is designed to meet all the functional requirements but is inefficient and insecure and therefore fails to satisfy the needs of its customers despite meeting their functional requirements. Hence, the system’s software architecture must be designed in such way that it should equally satisfy the system level NFR’s. However, this task is complicated by the presence of NFR conflicts, where design decisions made to support the resolution of one NFR has a detrimental effect on other NFRs. These NFR conflicts are analyzed with respect to their impact on each architectural element and the conflicts are resolved using the intelligent argumentation system Please refer to [16] for in detail explanation of NFR conflict analysis and detection in case of product lines. Please refer to [17] to understand more about NFR decomposition with respect to functional requirements.

V. CASE STUDY

In order to demonstrate the effectiveness of proposed method, a case study is performed on the architecture of our own intelligent argumentation system. This case study is performed in a simulated environment where we enacted ourselves as having different roles in software organization and participated in the design decision making process by posting arguments from various perspectives. The purpose of this case study is to capture the rationale behind the design decisions taken to design the architecture of the intelligent argumentation system. During the design of the system’s architecture, there were many design decisions made to address various design issues related to requirements which affected multiple architectural elements. One such issue is related to a requirement where users of the system need a tool to draw and describe their ideas using their own drawing patterns. This design issue can be resolved considering various design alternatives. The rationale based on arguments for the aforementioned design issue is captured by nine different stakeholders, each having various roles in software development. A decision is made based on the result provided by the intelligent argumentation system after the inference process.

The intelligent argumentation system is built on Client - Server architecture with distinction among presentation layer, business layer and Data persistence layer. The presentation layer handles user interface (UI) of the system that allows stakeholders to interact with the system and post their arguments. The business logic is handled by business layer where the inference process is performed. The data persistence layer deals with reading and writing the data into the database. This architecture consists of many components which are connected and communicating, through connectors. The architectural elements have their own rationale with respect to the design issues and the system is designed by resolving the issues with the help of intelligent argumentation system.

The stakeholders of the system, each having their own role, had provided their arguments to support or attack their choice of design alternative to address the design issue. Table I shows the various roles of the stakeholders in the organization who participated in the decision making process.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Role ID</th>
<th>Roles</th>
<th>Stakeholder Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D1</td>
<td>Developer</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>D2</td>
<td>Developer</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>U1</td>
<td>Participants/Users of System</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>U2</td>
<td>Participants/Users of System</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>U3</td>
<td>Participants/Users of System</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>AD1</td>
<td>Architecture Designer</td>
<td>0.9</td>
</tr>
<tr>
<td>7</td>
<td>T1</td>
<td>Maintainer/Tester</td>
<td>0.7</td>
</tr>
<tr>
<td>8</td>
<td>T2</td>
<td>Maintainer/Tester</td>
<td>0.7</td>
</tr>
<tr>
<td>9</td>
<td>S1</td>
<td>Sales Representative</td>
<td>0.8</td>
</tr>
</tbody>
</table>

For the design issue mentioned above, there are three design alternatives which can address the issue in their own way, canvas integrated in system, standalone drawing tool, and
desktop sharing. By integrating the canvas into the system, users can draw and share the same with other users without leaving the current session. The standalone drawing tool can also be used to draw patterns but it is not integrated in the current system forces users to toggle between multiple applications. Using desktop sharing, users can expose their drawings to others and share their ideas. With these three design alternatives, the stakeholders participated in the argumentation either supporting or attacking the design alternative or arguments posted by others. One of the components, issue based architecture design rationale capture as explained in the above framework is responsible to capture the stakeholder’s rationale. Table II shows the arguments posted by the stakeholders for one of the winning design alternative, Canvas Integrated in the system. Argument Id A1.1 depicts that this argument is posted under another argument having id A1.

Similarly the stakeholders posted their arguments for the other two design alternatives. After the inference process is carried out based on the stakeholder’s arguments, the system computed that the integrated canvas is the design alternative that is most favored by stakeholders. Based on the computed decision, a new component canvas is introduced into the architecture of the Intelligent Argumentation System. This new component also interacts with other components in the system. But, there is no direct impact on other architectural elements with the introduction of canvas. For this component to interact with other dependent components, a connector is introduced. Hence, the rationale for the architectural component, Canvas is captured and recorded in the knowledge base. In similar manner, all other design issues are resolved during architecture design and their rationale is recorded within a permanent repository of architectural knowledge. The documented architectural knowledge consists of the business requirement, the design issue, design alternatives, the entire stakeholder’s argumentation and the design decision. The component, software architecture design rationale knowledge base of the framework maintains the link between the architectural elements and their respective knowledge. Figure 5 shows the resulting architecture of the Intelligent Argumentation System.

The argumentation system architecture shows three high level components: Client, Server and Database system. These high level components are further classified into lower level components to resolve lower level design issues.

### Table II. Design Alternative (Position 1): Canvas Integrated in the System

<table>
<thead>
<tr>
<th>Argument ID</th>
<th>Stakeholder</th>
<th>Argument</th>
<th>Strength of the Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>D1</td>
<td>An implementation of a simple canvas with basic drawing capability involves less development and design</td>
<td>0.6</td>
</tr>
<tr>
<td>A1.1</td>
<td>D2</td>
<td>A simple canvas cannot give users more options to represent their ideas</td>
<td>-0.5</td>
</tr>
<tr>
<td>A1.1.1</td>
<td>U2</td>
<td>The users has to draw everything free hand. If we have already developed drawing patterns to draw a square, rectangle or cycle will help the user to spend less time on drawing</td>
<td>0.4</td>
</tr>
<tr>
<td>A1.1.1.1</td>
<td>D1</td>
<td>Since, the basis of our system is on argumentation and not on drawing, simple hand drawn images will suffice the requirement to represent the ideas pictorially</td>
<td>-0.8</td>
</tr>
<tr>
<td>A1.1.1.1.1</td>
<td>D2</td>
<td>The sophisticated drawing tool will enhance the representation of ideas and also decreases the human effort to hand draw all the images</td>
<td>-0.4</td>
</tr>
<tr>
<td>A2</td>
<td>U1</td>
<td>The canvas is simple to use and very easy for a user to understand and use the same</td>
<td>0.5</td>
</tr>
<tr>
<td>A3</td>
<td>D1</td>
<td>Since this canvas can be implemented in line with existing system, there is no extra development needed for integration as well as for sharing among the users</td>
<td>0.9</td>
</tr>
<tr>
<td>A3.1</td>
<td>AD1</td>
<td>This also decreases the complexity of the architecture and also it is a light weight application to be implemented</td>
<td>0.4</td>
</tr>
<tr>
<td>A4</td>
<td>S1</td>
<td>The use of canvas does not involve any licensing issue and also the existing computer’s hardware is suffice to meet the new requirement</td>
<td>0.5</td>
</tr>
<tr>
<td>A5</td>
<td>T2</td>
<td>This can be tested easily as it is part of our existing system and developed from scratch</td>
<td>0.6</td>
</tr>
<tr>
<td>A5.1</td>
<td>T1</td>
<td>This involves lot of test cases to be considered and increases the project duration</td>
<td>-0.5</td>
</tr>
</tbody>
</table>

![Figure 5 Architecture of Intelligent Argumentation System](image-url)
VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a method to resolve the architectural design issues and capture design rationale in the design of software architecture through intelligent argumentation. The design issues are resolved with the help of dynamic interactions among the stakeholders in terms of design issues, design alternatives, arguments, and evidences. The collaborative design decisions aid the system architects to design and modify the architecture of the system. This method captures the rationale in a structured manner and maintains the record of the architectural knowledge. The system is evaluated using a case study to validate the effectiveness of the method. This method is especially valuable in collaborative decision making environment where stakeholders are located geographically across different locations. If multiple products encounter same design issues, then the design knowledge captured for similar products can be retrieved and reused so that significant effort and cost can be saved in software design process. In future, we will develop a method to reuse captured architectural rationale knowledge across products to resolve similar design issues.

VII. ACKNOWLEDGMENT

We thank the Intelligent Systems Center at Missouri University of Science and Technology for supporting this project.

REFERENCES