Spring 2015

Argumentation based collaborative software architecture design and intelligent analysis of software architecture rationale

NagaPrashanth Chanda

Department:

ARGUMENTATION BASED COLLABORATIVE SOFTWARE ARCHITECTURE DESIGN AND INTELLIGENT ANALYSIS OF SOFTWARE ARCHITECTURE RATIONALE

by

NAGAPRASHANTH CHANDA

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2015

Approved by

Dr. Xiaoqing (Frank) Liu, Advisor
Dr. Maggie Cheng
Dr. Dan Lin

ABSTRACT

A growing model for software architecture defines it as a set of principle design decisions which drive system architects to design the architecture satisfying software requirements and architectural constraints. The design decision making process involves a group of stakeholders exchanging their viewpoints to address various concerns and to reach a consensus collaboratively. These 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 need to understand the rationale behind decision of each architectural element. This justifies the system design and avoids any critical architectural problems in future due to volatile requirements. Often, the architecture rationale behind various design decisions is not fully captured and hence affects the maintainability of software systems. In order to address the above research challenge, we developed an online intelligent software architecture rationale capture system (ISARCS) that enables stakeholders located at various geographical locations to resolve a design issue and capture the rationale behind issue resolution. The system captures a structured design rationale which maintains its links to software requirements and architectural elements.

This thesis also focuses on analyzing the architecture rationale captured during stakeholders discussion in various perspectives to provide stakeholders with a more detailed view that aids them in decision making. We propose use of intelligent argumentation analysis and various data mining techniques to analyze the software architecture rationale to unearth interesting information. Finally, a comprehensive empirical study is presented along with its experimental results.

ACKNOWLEDGMENTS

I would like to express my gratitude to all those who have helped me during this research. Firstly, I would like to thank my advisor Dr. Xiaoqing (Frank) Liu for giving me the opportunity to work on this project, his valuable insights and suggestions have helped me to overcome many hurdles during this work. I am grateful to him for the advice and guidance he gave me throughout my Masters program. I also thank Dr. Xiaoqing (Frank) Liu for his moral support during my tough times by understanding and encouraging me to focus on my goals. Further, I thank the Intelligent Systems Center for most of the funding during my Masters program. I would also like to thank Dr. Maggie Cheng and Dr. Dan Lin for being part of my thesis committee and taking time to review this work.

A special thanks to all my friends whose constant support and encouragement has always been crucial for me. Finally, I would like to dedicate this work to my parents (Mr. Sheshagiri and Mrs. Swaroopa) and my siblings (Mr.NagaPraneeth Chanda and Mrs. NagaPrasanna Choudhary) for their emotional support and unconditional love and sacrifices.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

1. INTRODUCTION

Software development is currently seen as an iterative process [1]. There are many phases involved in designing a software product following various software engineering methodologies as part of software development life cycle. The software requirements collected during requirements gathering phase play a key role in developing the software product in later phases of development cycle. The software products developed should fulfill the customer requirements to the maximum extent.

During one of the phases, based on the software requirements, the software architects design the architecture for the system. The software architecture design plays a vital role in software development since it provides framework of system for later development activities. The design of software architecture involves a decision making process to reach to a consensus by various design architects. During the design, the architects should make sure that the design decisions that they have taken are in congruence with software requirements. The mistakes made in architecture design have a significant impact on final product that is developed. Hence, the architects and other stakeholders of an organization should spend significant amount of effort during architecture design to avoid any defects in future during development.

The design of software architecture involves design decisions that are taken by design architects in order to fulfill various software requirements. The selection of architecture elements and the justification of design decisions are found in architecture rationale [2]. These set of principle design decision which describe software architecture [3], drives architects to develop an architecture that satisfies and reflects the software requirements. The main idea for representing design decisions is to bridge gap between software requirements and architectural products [4].

The requirements play a significant role in designing software architecture.

These requirements define what the system should do, whereas software architecture describes how this is achieved [1]. In order to fulfill the customer requirements, the software architecture can be designed in multiple ways. The software architects should analyze tradeoffs between conflicting requirements and make a design decision accordingly. The software architecture has been considered as a structure composed of components, connectors, constraints and rationale [5]. The functionality depicted in the architecture in terms of components and connectors should adhere to the software requirements.

The software requirements are main pillars to build software architecture. There are many design issues that arise in order to fulfill the stated requirements. The design decisions that are taken during resolution of a design issue have their own respective rationales which are often not documented as part of software artifacts. Most of the time, the rationale for design is limited to remain only in the brains of the designers. Therefore, due to the absence of the software architecture rationale, the architects face difficulties to modify the architecture of system as they are not able to assess the impact of changes on existing requirements and previous trade off decisions. In order to effectively develop a software product based on software architecture, the rationale should be captured and maintained and thus improving the understandability of software architecture. Hence, we can couple the software architecture design with its corresponding rationale and improve the efficiency of software development. By maintaining the rationale, architects can now trace back the software architecture elements to the design decision as well to their related software requirements.

1.1. SOFTWARE ARCHITECTURE RATIONALE CAPTURE

The satisfaction of software requirements is regarded as a design issue. There are many design solutions called design alternatives which can be chosen as a design in order to resolve a design issue. Each design alternative has its own pros and cons in

satisfying the requirements to an extent. The pros and cons of each design alternative have to be analyzed in various perspectives in their level of satisfying the related requirements. In order to resolve such a design issue, the design alternatives have to be analyzed by various stakeholders of an organization. But, different stakeholders may have different viewpoints in order to address the design issue. Some of the viewpoints may be contradictory while some of them might be similar. The viewpoints of all the stakeholders needs to be considered and analyzed in order to reach to a consensus that addresses the concerns of entire group. Generally, architects design the software architecture based on their experience and knowledge but sometimes their knowledge is limited and may not provide a fruitful end result. So, the stakeholders need to collaboratively exchange their views and thus build an organized rationale that enables them to reach to a most viable design solution.

An online intelligent software architecture rationale capture system (ISARCS) is developed that enable stakeholders from various geographical locations to participate in an online discussion to share their viewpoints in terms of arguments in order to resolve a design issue. Stakeholders can either attack or support a design alternative that resolves a design issue. The entire argumentation maintains its link to related software requirements and software architectural elements. The argumentation rationale captured during online discussion, serves like a justification for evolution of software architecture. A huge argumentation tree that is built during design issue resolution is analyzed to identify the most favored alternative by most of the stakeholders.

1.2. ANALYSIS OF SOFTWARE ARCHITECTURE RATIONALE

The argumentation data that is captured during stakeholders discussion is analyzed using various data mining techniques in different perspectives to provide stakeholders with a more comprehensive view that supports them in selecting a best

design solution.

After the argumentation process, the data is analyzed using various approaches to classify the arguments, to develop a traceability matrix based on rationale and to perform topic analysis. The stakeholders viewpoints can also be analyzed to detect the impact of selection of a design alternative based on various software concerns.

On other hand, since the discussion is spanned for several days. The data that is captured on each day of argumentation can be analyzed to unearth some interesting information. Different evolutionary approaches are proposed to show the convergence of discussion to reach to a consensus, to detect arguments that have changed some of the stakeholders opinions, to detect the participation of stakeholders in discussion and to detect arguments which have gained significant attention during discussion.

1.3. ORGANIZATION OF THESIS

The rest of the thesis is organized as follows. Section 2 presents a review about the related work in field of software architecture management. In Section 3 details about background of Intelligent Argumentation System. Further, Section 4 explains in detail about the process employed to capture software architecture rationale using ISARCS. In Section 5, intelligent analysis of architecture rationale is discussed. Later, in Section 6, we explain about various approaches that are used to perform evolutionary analysis on architecture rationale. Finally, in Section 7, we talk about the empirical evaluation of a case study which exemplifies different approaches proposed with experimental results.

2. RELATED WORK

This section presents the literature review of related research. 2.1 presents the work done in field of architectural knowledge management and 2.2 discusses the related work of other argumentation systems.

2.1. ARCHITECTURE KNOWLEDGE MANAGEMENT

There has been a significant amount of work in architectural knowledge management in order to bridge gap between software architecture and architecture design decisions. In one of the approaches proposed by Fabian et al [6], design decision 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. Van et al [1] proposed an approach to bridge gap between rationale and architecture artifacts by uniting them into a concept of a design decision. In another approach, Perry et al [7] considers the management of architecture design decision an important aspect in case of global software development. Also, they consider that the rationale provides a design decision that serves as a justification for the architectural elements. Capilla et al [8] proposes a web based tool, ADDSS, for documenting design decision during architecture design. The proposed approach connects requirements to architectures via design decisions and enables traceability between them. ADDSS captures rationale by linking motivating factor to design decision made, also stores design pattern knowledge. PAKME [9] captures design alternatives as cases from literature. It considers that a design case consists of problem and solution, patterns and tactics used, rationale and related design options. Cui et al [10] proposed a design centric architectural design in which stakeholders determine the architectural issue

from requirements as well as their solutions. The system explores all the feasible combinations of the issue solutions and combines the feasible combinations to generate architectural solution. They as well measure the pros and cons of each solution to generate an architectural solution. Architecture as design decisions was presented by Lytra et al [11]. The system automates component and constraint generation based on design decisions. Each design decision has a set of outcomes, which are mapped directly to architectural elements they generated in the component model, allowing for each component to be traced back to its corresponding design decision. Savolainen and Mannisto [12] proposed a method of creating architectural views that more prominently communicate the conflicts from multiple perspectives between key stakeholders concerns. It helps in capturing architectural rationale based on interactions among stakeholders. Bratthall et al [13] propose the importance of design rationale in change management and also an approach to document the architectural knowledge to verify change impact.

All the above methods discussed states the importance of maintaining architecture rationale for software architecture management. But, they fail to propose a method to capture the rationale by consulting various stakeholders of organization. Apart from that, the rationale captured by above methods are not structured and organized in terms of arguments, requirements and other architectural elements. The methods also fail to analyze the architectural rationale from multiple perspectives to fetch more information that helps stakeholders to understand the design issue and design decision in detail.

2.2. ARGUMENTATION SYSTEMS

There is also a large amount of work done in collaborative decision making through argumentation. Most of the argumentation systems proposed follow Stephen Toulmins model of argumentation [14]. The first method was gIBIS [15] which repre-

sents design dialog as a graph. The method displays arguments, issues, and positions in the form of a graph. HERMES [16] is a computer based decision support tool which organizes arguments and evidences in hierarchy. Chenn-Junn Huangs [17] argumentation system assess the quality of the arguments by parsing arguments.

These argumentation systems discussed above serves as a decision support system and are biased to general issues and are not related to software architecture design. They fail in capturing rationale for a software architecture design as the resolution of a design issues involves consideration of other elements like requirements and architecture design.

3. BACKGROUND

As part of preliminary work , Intelligent Argumentation System was proposed to capture software architecture rationale [18]. But, later the idea was enhanced to develop a new system called ISARCS, that can be used to capture software architecture rationale in a more structured and organized way. The new system, ISARCS is built on existing Intelligent Argumentation System that is used for collaborative decision support and has displayed promising results.

3.1. INTELLIGENT ARGUMENTATION SYSTEM

Different people have varied opinions and views with respect to the issue which contradict to each other in many ways. Generally issues may be related to an organization like introduction of new business strategies or in a software development process like architecture design or to global challenges like global warming, national debt or genetic modified crops. Whenever there are such issues, people discuss a lot about them in course of time and as well attract significant attention of other people across the globe. Our system allows users to collaboratively participate and resolve an issue through online discussion. As part of argumentation system, the issue is referred as an *Issue* and various alternatives that address the issue are termed as *Positions*. In order to resolve the issue, the users exchange their views in terms of arguments supporting or attacking a position. The users 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 arguments with positive degree signify that it is supporting another argument. An argument with 0 degree signifies indecisiveness [19]. The priority of the users is also one of the factors in decision making. So, the users are assigned priorities based on their expertise and

knowledge of topic [20].

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 as shown in Figure 3.1. 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 [20, 21, 22, 23]. 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.

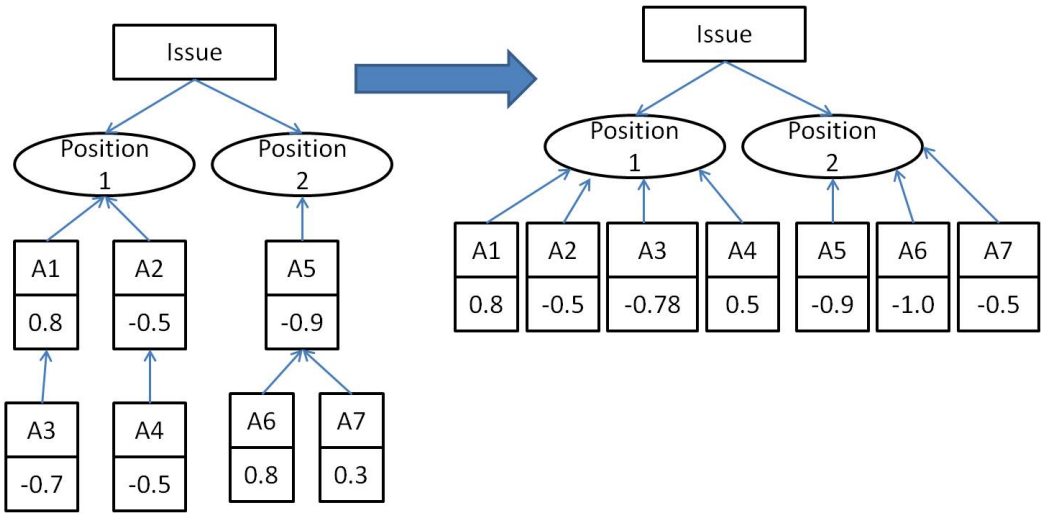

Figure 3.1. Sample Argumentation Tree Reduction

4. ISARCS : INTELLIGENT SOFTWARE ARCHITECTURE RATIONALE CAPTURE SYSTEM

In this section, we introduce our system ISARCS that aids organizations to resolve software architecture design issues collaboratively and develop an architecture that is favored by most of the stakeholders. In order to capture and maintain the architecture knowledge [24], we have developed a web based tool that enables stakeholders located at various geographical locations to participate in an online discussion to resolve a design issue. It is built on client-server architecture where client provides users an interface to interact with the system and server analyses and maintains different clients connected to system. The entire discussion is captured and maintained and thus provides a permanent repository of architecture knowledge.

Software requirements are basis for architecture design. Whenever a design decision is chosen, the decision should satisfy the requirements to maximum extent. Apart from software requirements, there are architectural constraints that should as well be satisfied by the design decision. Figure 4.1 shows the architecture design flow where ISARCS is used to resolve a design issue that arises from software requirements and existing architectural elements. The design issue can be resolved with various design alternatives. But, not all alternatives have same impact on existing architecture or may have varied levels of satisfaction of software requirements. The stakeholders, using ISARCS discuss the pros and cons of each design alternatives with respect to related requirements and architecture elements. The system analyzes the discussion using data analysis engine to provide stakeholders with a more comprehensive view of discussion. The architecture knowledge maintained using ISARCS, helps architects or stakeholders in design of architecture in future, if they want to revisit the architecture due to change requirements.

The ISARCS is built on existing Intelligent Argumentation System [20, 21,

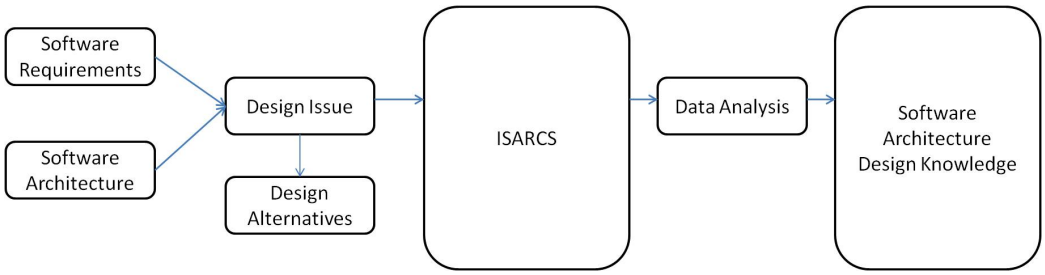

Figure 4.1. Software Architecture Design Flow

22, 23] which displayed promising results in resolution of general issues. During software development, there are numerous requirements gathered during requirements gathering phase. Some of the requirements may result in certain design issues which needs to be resolved to design the architecture for a system. Since, not all requirements captured are responsible for a design issue, the stated design issue is mapped to relevant requirements list. As part of ISARCS, we capture the relationship between a design issue and also the requirements that are related to a design issue. Apart from requirements, sometimes a design issue might impact existing architectural elements. In that case, the system will also capture the elements which might be impacted due to resolution of a design issue. Figure 4.2 shows the relationship of software requirements and existing architectural elements to design issue. Once a design issue is proposed, there are many ways to resolve the design issue using design alternatives. These design alternatives are as well mapped to the software requirements, architectural elements and architectural constraints. The clear mapping of related requirements and existing architectural elements enables stakeholders to understand various relationships before they participate in online discussion to resolve issue collaboratively.

Figure 4.3 shows the architecture of ISARCS. It mainly consists of four high level components which are connected to each other using connectors. The first component, collective discussion engine is responsible to collect the viewpoints of various

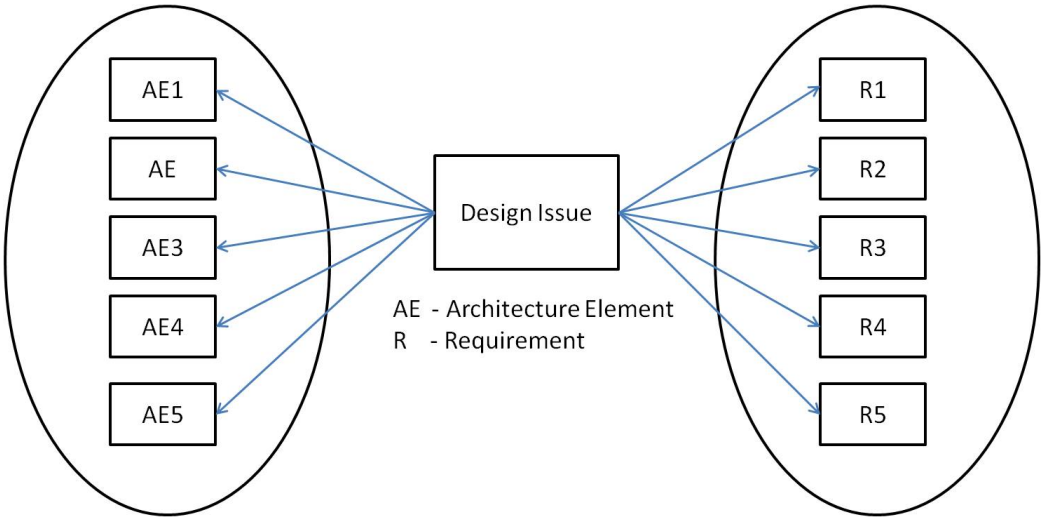

Figure 4.2. Design Issue Relationship

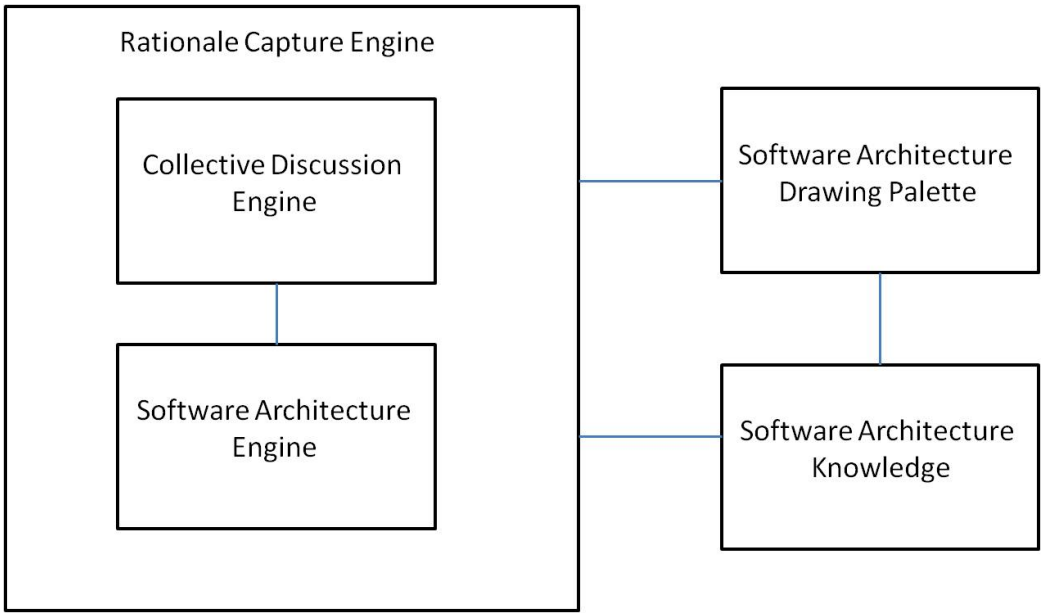

Figure 4.3. ISARCS Software Architecture

stakeholders regarding the design issue and as well maintain entire discussion chains. The component software architecture engine enables stakeholders to maintain software requirements, software architecture elements in system. The third component, software architecture drawing palette aids the stakeholders to draw the architecture during discussion and thus provides a visual view of software architecture under discussion. The software architecture knowledge component is a permanent repository of the entire architecture rationale which consists of stakeholders arguments, architectural elements, architecture design, software requirements and their inter relationships.

Figure 4.4 shows snapshot of ISARCS. The stakeholders will be able to participate in discussion stating their viewpoints in terms of arguments to support or attack various design alternatives. Apart from just posting their arguments, stakeholders can post evidences in order to strengthen their arguments. Whenever a stakeholders posts arguments, he can enter an argument weight in range of -1 to 1 to signify his degree of support or attack of a design alterative or with respect to another arguments. As the discussion evolves, a huge argumentation tree is built that is analyzed to compute a design decision that is favored by most of the stakeholders using argumentation reduction inference engine [23].

A stakeholder viewpoint might be related to a certain requirements or to certain architecture constraints or to some architectural elements. So, when a stakeholder is posting his viewpoint, he will be able to choose the related requirements, architectural elements that his argument is based on. In that case, other stakeholders in discussion can understand the basis of the argument posted by a stakeholder and can respond accordingly. The chosen requirements and architecture elements are subset of those that are linked to design issue as shown in Figure 4.5. The entire discussion of arguments, design issue and design alternatives are all linked to software requirements, architectural elements that provides stakeholders with clear understanding of

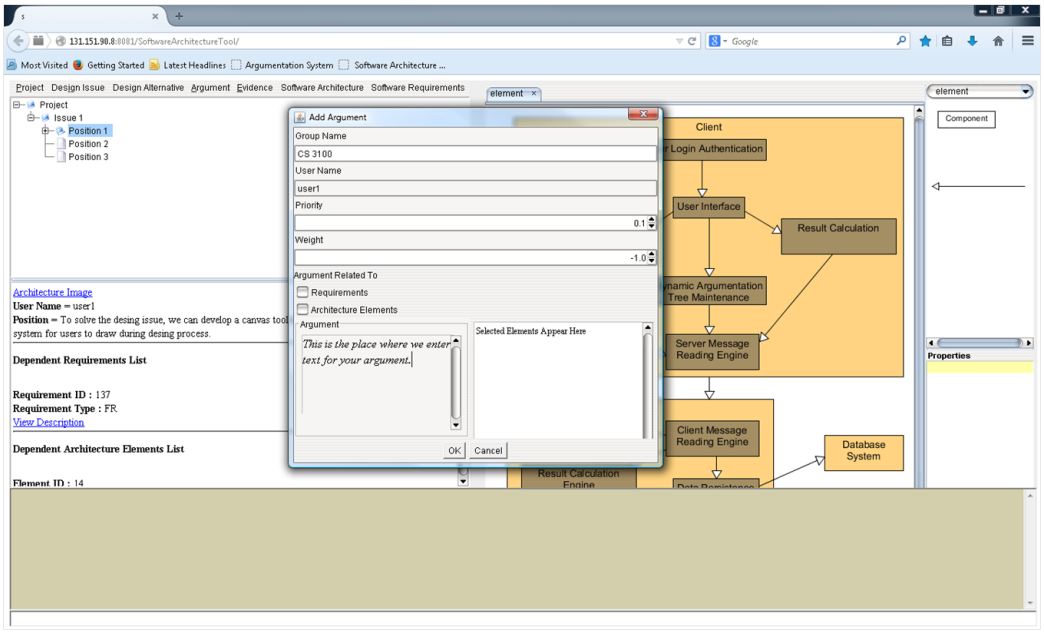

Figure 4.4. A Snapshot of ISARCS

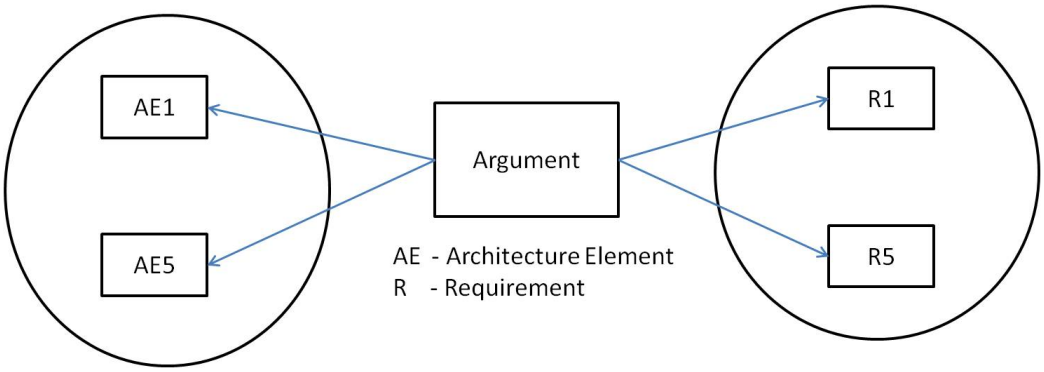

Figure 4.5. Argument Relationship

the relations among various elements of system. The system also provides a drawing area that enables stakeholders to draw the software architecture during online discussion to state their idea clearly using visual ques. The software architecture that is drawn is as well linked to a design issue and other design alternatives that provides stakeholders a view of how architecture is affected if a particular design alternative is selected.

5. INTELLIGENT ANALYSIS OF SOFTWARE ARCHITECTURE RATIONALE

In this section, we propose various approaches to analyze the software architecture rationale captured during design decision making process.

5.1. ANALYSIS OF SOFTWARE CONCERNS

In order to resolve a design issue, since many stakeholders state their viewpoints to support their design alternative, after discussion, there is a huge tree that is built and it is a difficult task for stakeholders to get an overview about entire discussion. To provide the stakeholders with an overview of discussion, we can analyze the arguments and provide them with an analysis to detect affect of various software concerns on individual design alternatives. The software concerns are generally extracted by manual analysis of software requirements. There are many concerns like cost, development effort, development time, security, software testing, software maintenance that can be considered to analyze the arguments and provide stakeholders with their effect on alternatives.

Algorithm 1 is employed to compute the effect of each concern on design alternatives. The input to the algorithm is a set of argument lists in the argumentation tree, a list of software concerns and a design alternative. After the argument analysis, the algorithm outputs a matrix that shows the effect of a concern on a design alternative. Lines 1-3 in algorithm reduces all the arguments in tree to a single level such that they directly point to their respective design alternatives using fuzzy logic based argumentation reduction inference engine. During the reduction, the strengths of the arguments are recalculated to compute their effect on a design alternative. At Line 5, we collect the synonyms of the concerns using thesaurus. Since, many stake-

Alg. 1 SoftwareConcernAnalysis(*ArgumentationTree*, *ConcernsList* , *DesignAlternative*)

Input: Argumentation Tree; A list of software concerns; Design Alternative
Output: Affect of Concerns on a design alternative

```
1   begin
2       for each arg in ArgumentationTree(DesignAlternative) do
3           argWeight = ArgumentationInferenceEngine(arg)
4       for each concern in ConcernsList do
5           synonymConcernsList = Synonyms(concern)
6           for each arg in ArgumentList do
7               for each concernItem in synonymConcernsList do
8                   if arg contains concernItem then
9                       overAllEffect = overAllEffect + argWeight
10  end
```

holders can express their viewpoint about a concern in various ways. So, we need to consider all the possible available words for a given concern. For Example: If we consider a software concern as cost, then we consider all the other words which are synonym to cost like expensive, expense, expenditure, money etc,. Once we determine all the required words for a concern, we parse through all the arguments to verify if the argument is talking about that particular concern. This is represented as part of line 6-8 in the algorithm. Once we detect all such arguments, we aggregate the recalculated strengths of those arguments to compute the overall effect of a concern on a design alternative. After the entire analysis, the algorithm provides stakeholders with a Software Concerns Effect Matrix as shown in Table5.1.

Table 5.1. Effect of Software Concerns

Software Concern	Design Alternative 1	Design Alternative 2	Design Alternative 3
Cost	Overall Related Arguments Effect	Overall Related Arguments Effect	Overall Related Arguments Effect

5.2. CLASSIFICATION OF STAKEHOLDERS VIEWPOINTS

During architecture design, individual stakeholders express their viewpoints in terms of arguments. The credibility or the influence of an argument can be analyzed by computing collective thoughts of other stakeholders on that argument. The assessment of collective thought on an argument conveys more information from a group's perspective. By determining the collective opinion of a group on an argument posted by a stakeholder, we can detect arguments that have gained more attention in discussion. The collective weight of other stakeholders arguments provides more insight to determine arguments that attracted more support from other stakeholders and also arguments that were attacked by most of the stakeholders.

As said earlier that during argumentation, stakeholders associate a weight of -1 to 1 to an argument. The collective thought on argument is computed by considering the arguments posted by stakeholders either supporting or attacking it. The collective thought which is termed as collective determination of an argument is summation of collective support and collective attack an argument receives. The collective support an argument is calculated based on arguments that support it and collective attack for an argument is determined considering arguments that attack it. The collective determination value of an argument determines the arguments that have contributed significantly in selecting a design alternative.

$$CollectiveDetermination = CollectiveSupport + CollectiveAttack$$

Figure 5.1 shows a sample argumentation tree consisting of eight arguments posted under a design alternative. The collective determination for argument A2 is computed by considering arguments that refer A2 either directly or indirectly. In this case, arguments A5, A6, A7 and A8 refer to argument A2 and hence these arguments

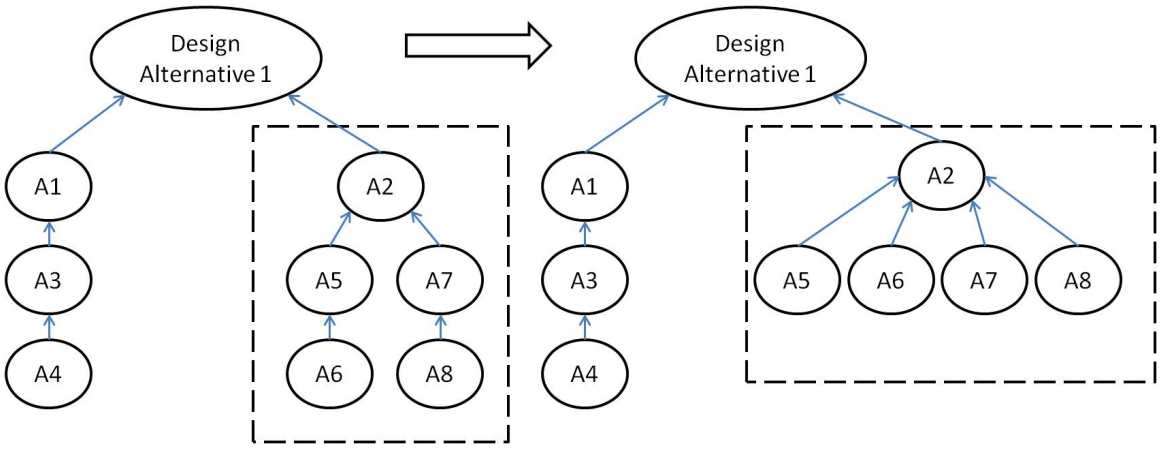

Figure 5.1. Sample Argumentation Tree for Collective Opinion Analysis

contribute to compute collective determination of A2. As arguments A6 and A8 refer argument A2 indirectly, we need to compute the effect of both the arguments by reducing them to a single level pointing directly to A2. The reduction of indirect arguments to single level is performed using argumentation reduction fuzzy inference engine. For ex. In this case, the argumentation reduction engine takes strengths of arguments A5 and A6 as input and computes the effect of A6 on A2. Similarly, the effect of argument A8 on A2 is computed using A7 and A8 as inputs to inference engine. After the argumentation reduction, the collective support for A2 is computed by summation of arguments that support A2 and collective attack for A2 is computed by summation of arguments that attack A2.

The collective determination for all the arguments in the argumentation tree is computed to detect arguments that have contributed a significant weight and attracted more attention during discussion. The collective determination values for leaf arguments are considered as zero as they are not supported or attacked by any other arguments. Table5.2 shows classification of arguments based on collective opinion values and number of follow up arguments posted for an argument.

The arguments which are having more number of follow up arguments have

Table 5.2. Classification of Stakeholders Arguments

Condition	Argument Classification
MAX(Number of Follow up Arguments)	Attention Seeking Arguments
MAX(Collective Support)	Most Supported Argument
MIN(Collective Attack)	Most Attacked Argument
MAX(Collective Determination) or MIN(Collective Determination)	Decision Centric Arguments

attracted significant number of stakeholders to participate in discussion. Such arguments have created a buzz in discussion and are classified as *Attention Seeking Arguments*. There are also some arguments which receive a lot of support from other stakeholders under discussion. The support an argument receives may be attributed to its authenticity and credibility and such arguments are classified as *Most Supported Argument*. On other hand, there are arguments that are attacked by most of the stakeholders, in such case, most of the stakeholders disagree with the argument because of the conflicting nature of the argument. Such arguments are classified as *Most Attacked Arguments*. When we consider the collective determination value of arguments, the arguments that have maximum and minimum values will impact the final weight on design alternative and such arguments might result in deciding the final design decision. So, those arguments are classified as *Decision Centric Arguments* as they are crucial in choosing a design solution.

5.3. SOFTWARE REQUIREMENT TRACEABILITY

In software architecture design, every architectural element in the architecture is associated with its related software requirements. The traceability between software requirements and architecture is of critical importance as it helps the system architect to analyze the impact of volatile requirements on existing architecture elements. On other hand, an architect can also analyze the requirements that are affected if any of

the architectural elements is modified.

As part of ISARCS, every design issue stated is mapped to its related software requirements so that stakeholders are cognizant of the requirements that they are dealing with during online discussion. The resolution of a design issue might result in a new architectural element in architecture or might have modified existing architectural elements. As software systems evolve, there are many design issues that needs to be resolved. Every design issue is related to a subset of software requirements. The resolution of the design issues impacts the software architecture design. Since, the modifications of the architecture of system is through design issue resolution, and the design issue is mapped to software requirements, the design solutions which reflects software architecture are as well mapped to the respective requirements lists as shown is Figure 5.2.

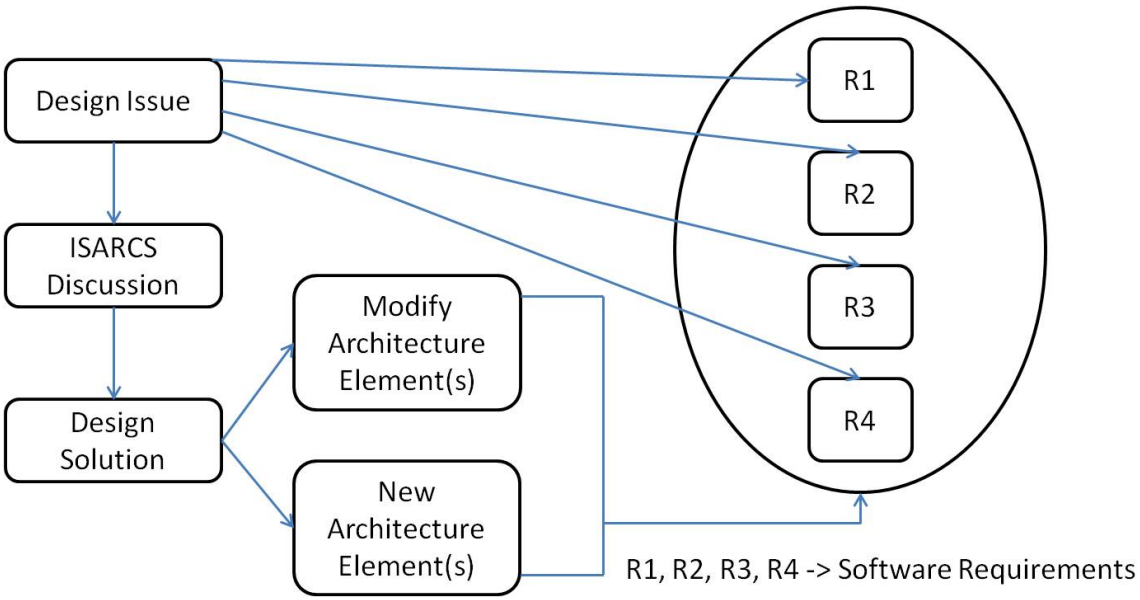

Figure 5.2. Traceability of Requirements and Architectural Elements

Table 5.3 shows sample design issues, *Issue 1* to *Issue 4* during architecture design which are resolved using ISARCS to compute different solutions, *Solution 1* to

Table 5.3. Relationship among Issues, Requirements and Architecture Elements

Design Issue	Related Requirements	Design Solution	Impact on Architecture
Issue 1	R1, R5, R6	Solution 1	New Architecture Elements (AE1, AE2)
Issue 2	R1, R4, R7	Solution 2	Modified Architecture Element (AE4)
Issue 3	R1, R8, R7	Solution 3	New Architecture Element (AE3)
Issue 4	R3, R1, R2, R8	Solution 4	Modified Architecture Elements (AE5, AE6)

Solution 4. We can see that each design issue is related to a subset of requirements. Some of the requirements might also be part of multiple design issues. Each design solution has an impact on architecture, For Eg: the *solution 1* has yielded two new architecture elements AE1 and AE2 into software architecture. These two architecture elements AE1 and AE2 are now mapped to requirements R1, R5 and R6.

Similarly, architectural element AE4 which got modified due to *Solution 2* is mapped to requirements R1, R4 and R7. These requirements R1, R4 and R7 are appended to existing requirements lists which are already mapped to AE4. In future, if any of the requirements change says R2, then the architects now can trace back to architecture and determine that AE5 and AE6 are elements that need to be checked on. Similarly, if AE1 is modified for any reason, the architects needs to trace back to related requirements and verify that the requirements are not violated due to modified design.

Based on the above analysis, we developed a requirements traceability matrix which maps the software requirements to the software architecture elements. Table 5.4 shows a sample traceability matrix which depicts the relationships between architectural elements and software requirements. The numeric value 1 in cell illustrates that the architectural element is mapped to the corresponding requirement

and 0 represents that there is no relationship between them. When we observe the traceability matrix, we can see that requirement R1 is mapped to six different architectural elements and similarly architectural elements AE5 and AE6 are mapped to 4 different requirements. By analyzing the matrix, we can check the impact of modifying a requirement on architecture design and similarly can verify the impact of modifying the architecture design on software requirements. For Eg. If requirement R1 is modified in future, then it has a significant impact on software architecture as it affects 6 architectural elements and a major modification of design is involved. In that case, the stakeholders of the organization have to rethink of allowing changes to such requirements which might hinder there entire development process.

Table 5.4. Requirements Traceability Matrix

Requirements/ Architecture Elements	R1	R2	R3	R4	R5	R6	R7	R8	Total
AE1	1	0	0	0	1	1	0	0	3
AE2	1	0	0	0	1	1	0	0	3
AE3	1	0	0	1	0	0	1	0	3
AE4	1	0	0	0	0	0	1	1	3
AE5	1	1	1	0	0	0	0	1	4
AE6	1	1	1	0	0	0	0	1	4
Total	6	2	2	1	2	2	2	3	0

5.4. TOPIC ANALYSIS

In an online discussion, each stakeholder expresses his viewpoints in different perspectives for available design alternatives. But, as the online discussion evolves, there are numerous arguments posted by stakeholders which address various concerns of discussion. This results in tedious task to understand the state of argumentation and also its very difficult to study each argument individually to perceive the content posted. Of all the arguments, some of the arguments refer to a similar subject of

concern and identification of such concerns will enable the stakeholders to understand various topics that are under discussion and the topics which grasped large attention in discussion. In order to detect the topics of interest, the entire argumentation tree have to be analyzed and the arguments which talk about similar concerns have to be grouped to evaluate the strength of the concern.

We can make use of text clustering techniques to cluster the arguments and identify the topics that are often under discussion [25]. The clustering result provides stakeholders with an overall view of the topics that are most discussed. Clustering also enables to group arguments according to the concerns they represent. The text clustering of arguments is performed using Lingo algorithm [26] that is generally used to cluster web search results. Each cluster in the result is represented by a cluster label that denotes the topic of interest of the arguments present in that cluster. There is always a possibility that an argument might discuss about multiple topics. In that case, the argument is part of multiple clusters.

6. EVOLUTIONARY ANALYSIS OF SOFTWARE ARCHITECTURE RATIONALE

In this section, we propose various approaches to perform evolutionary analysis of the software architecture rationale captured during design decision making process.

6.1. DETECTION OF CONVERGENCE USING POLARIZATION GROUPS

The main idea in an online decision making is generally to reach to a consensus by various stakeholders sharing their viewpoints collaboratively. In architecture design process, ISARCS facilitates stakeholders to express their opinions in terms of viewpoints supporting or attacking various design alternatives. During decision making, stakeholders tend to form different polarization groups [27, 28]. The stakeholders with similar opinions tend to get closer by supporting one another. Similarly, stakeholders within a group have contrasting opinions with other polarization groups. The online discussion often influences certain stakeholders to change their opinion on design alternatives. As discussion is evolved, there is always a possibility that a stakeholder might attack an alternative that he was previously supporting. On other hand, he might also support an alternative that he might have attacked earlier. The change in decision can be attributed to influence of various arguments on his opinion.

Whenever a stakeholder gets influenced due to the discussion, he tends to move away from his earlier similar stakeholders group and move closer to other group. This makes stakeholders to jump from one polarization group to another. In order to study such transitions of stakeholders among different groups, the evolution of entire discussion on each day should be analyzed. The evolutionary analysis of online discussion aids in understanding the dynamics of argumentation. Initially the stakeholders are scattered around different groups. But, as the discussion is evolved each

day, stakeholders tend to understand the pros and cons of each design alternative and tend to form a group with significant number of stakeholders. Since, the idea for collaborative discussion is to reach to a consensus, during discussion we can see a large number of stakeholders forming a polarization group whose opinion decides the final design solution. Such evolutionary analysis helps in visualizing the convergence of stakeholders opinions in choosing a design alternative.

We propose a clustering process to form different polarization groups. As stated earlier, during online discussion, a stakeholder associates a weight to signify his degree of attack or support. We use the stakeholders favorability values for various design alternatives as basis to form cluster groups. The stakeholders favorability implies his degree of support and degree of attack on various design alternatives. The overall favorability of a stakeholder on a design alternative is either positive or negative. The cluster groups are decided based on number of design alternatives available. So, if there are two design alternatives, then there would be 4 cluster groups formed. Similarly, if there are 3 design alternatives, the number of cluster groups considered is 9 as shown in Tables 6.16.2

Table 6.1. Cluster Groups For Two Design Alternatives

Cluster Group	Overall Favorability(Design Alternative 1, Design Alternative 2)
C1	*(Positive , Positive)*
C2	*(Positive, Negative)*
C3	*(Negative, Positive)*
C4	*(Negative, Negative)*

Based on the tables, for two design alternatives, if the overall favorability of a stakeholder on design alternative 1 is positive and on design alternative 2 is negative then he would belong to cluster group C2. Similarly, for three design alternatives, if overall favorability of a stakeholder on alternative 1 is positive, alterative 2 is

Table 6.2. Cluster Groups For Three Design Alternatives

Cluster Group	Overall Favorability(Design Alternative 1, Design Alternative 2, Design Alternative 3)
C1	*(Positive, Positive, Positive)*
C2	*(Positive, Positive, Negative)*
C3	*(Positive, Negative, Negative)*
C4	*(Positive, Negative, Positive)*
C5	*(Negative, Positive, Positive)*
C6	*(Negative, Positive , Negative)*
C7	*(Negative, Negative, Positive)*
C8	*(Negative, Negative, Negative)*

negative and alternative 3 is positive then he would belong to cluster group C4. The favorability values of each stakeholder who participated in online discussion is computed and assigned to respective clusters based on the determined values. If we consider cluster group C7 for three design alternatives, then all the stakeholders who are present in C7 collaboratively attack alternatives 1, 2 whereas support alternative 3. On other hand, the stakeholders in cluster group C8 attack all the three design alternatives and agree with neither of them.

Let us exemplify by considering sample argumentation tree consisting of two design alternatives and two stakeholders posting arguments supporting or attacking them as shown in Figure 6.1.

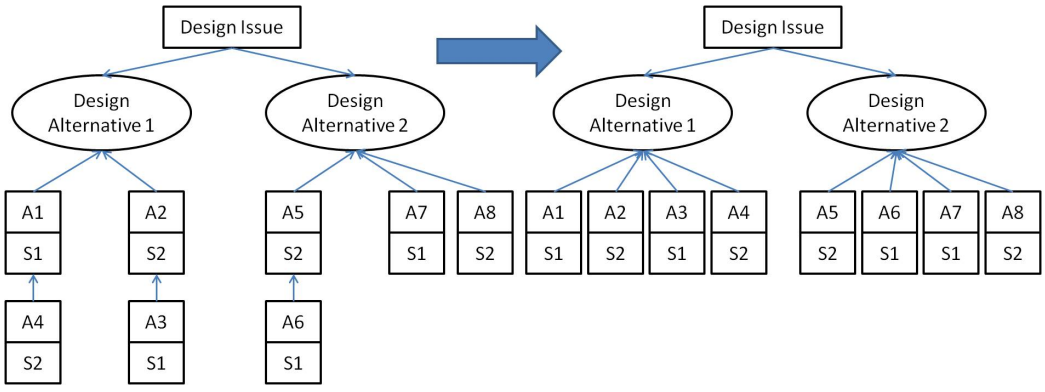

Figure 6.1. Sample Argumentation Tree to Determine Stakeholder Favorability

Stakeholder S1 posted one argument A1 directly on design alternative 1 and argument A3 on argument A2 which refer indirectly to design alternative 1. Similarly, S1 posted arguments A7 and A6 which refer to design alternative 2. On other hand, stakeholder S2 posted arguments A2, A4 on design alternative 1 and A5, A8 on design alternative 2. The arguments like A4, A3, and A6 which refer indirectly to design alternatives are reduced using argumentation reduction inference engine to compute their effect directly on design alternatives. Each argument in the argumentation tree is associated with weights signifying their degree of support or attack. For stakeholder S1, the overall weight of his support or attack on a design alternative is computed by weighted summation of arguments posted under that alternative. In this case, for design alternative 1, the summation of arguments weights A1 and A3 determines his decision weight. Similarly, for stakeholder 2, the summation of weights A2 and A4 determines his weight of support or attack on design alternative 1. The opinion of each stakeholder is represented by a tuple where each element in tuple determines the decision weight on design alternatives. In this example, stakeholders S1 and S2 each have a tuple, say, (S1W1, S1W2) and (S2W1, S2W2). Based on the signs of elements in tuple, the stakeholders are assigned to respective cluster groups. If S1W1 is positive and S1W2 is negative then S1 would belong to cluster group C2. Similarly, if S2W1 negative and S2W2 are positive, S2 is associated with group C3. The tuples for all the stakeholders in the discussion are calculated and are assigned to their respective cluster groups forming clusters.

$$S1W1 = Weight(A1) + Weight(A3),$$

$$S1W2 = Weight(A6) + Weight(A7)$$

$$S2W1 = Weight(A2) + Weight(A4),$$

$$S2W2 = Weight(A8) + Weight(A5)$$

As the discussion is spanned for several days, the stakeholders who participated in discussion are clustered into different cluster groups on each day. The clustering process is performed on each day of discussion. Every day some new stakeholders join into discussion and some stakeholders might change their opinion on certain design alternatives. When we cluster the stakeholders opinions on each day, we can visualize that certain cluster groups gain more number of stakeholders into their group where as some groups tend to lose their strength. As the discussion is evolved, certain cluster groups becomes more and more large and others shrink in their number and after final day of discussion, we can see that a cluster group has more number of stakeholders when compared to other groups. The size of cluster group signifies the design solution that is favored by most of the stakeholders. For Example: If there is an architecture design issue with two design alternatives and the discussion is carried out for 3 days with 10 stakeholders. On final day, if 6 stakeholders are present in cluster group C2, it signifies that the most of the stakeholders support design alternative 1 and attack design alternative 2 suggesting a convergence point in the decision making process.

6.2. DETECTION OF INFLUENTIAL ARGUMENT CHAINS

As stated earlier, that during decision making process, some of the stakeholders might change their opinion and move from one cluster group to another cluster group. But, what has made those stakeholders to change their cluster group? Generally in any discussion, different people share their ideas about their likes and dislikes. Some of the ideas receive more attention and people tend to discuss a lot about them

forming a deep discussion chain. Such discussion chains always gains significant attention and may influence an individual to opt the idea discussed. In ISARCS, as well stakeholders discuss in resolving a design issue by exchanging their rationale in supporting or attacking various design alternatives. Certain arguments gain a lot of attention resulting in many numbers of arguments following it forming a deep argumentation chain. These branches of argumentation chains in an argumentation tree tend to influence certain stakeholders to change their opinion on a design alternative.

We can detect such influential argument chains in tree by studying the clustering evolution of discussion. A stakeholder during discussion is associated with a cluster group along with other similar stakeholders. But, when he encounters such argumentation chains and changes his decision then he would start supporting an alternative that he attacked earlier and attacks other alternatives. When he changes his decision then his overall favorability on an alternative might either increase gradually to reach positive or might decrease gradually to reach negative weight. In that case, the elements in the stakeholders tuple change their signs and hence stakeholder is assigned to different cluster group.

Consider cluster evolution of stakeholder S1 for a period of 6 days as shown in Figure 6.2. During discussion, we can see that stakeholder S1 was part of cluster group C2 supporting design alternative 1 and attacking design alternative 2 for first 3 days. But, later from day 4, S1 started attacking design alternative 1 and supporting design alternative 2 and changing his cluster group to C3 from C2. The reason for the stakeholders change in his opinion can be attributed to various arguments posted by other stakeholders. As stated earlier that some of the argument chains might have influenced the stakeholder to change his opinion. The argumentation tree for each day needs to be analyzed to detect from which day there is a decrease in opinion weight of S1 on design alternative 1 and increase in opinion weight on design alternative 2.

Consider a sample argumentation tree as shown in Figure 6.3. In figure, el-

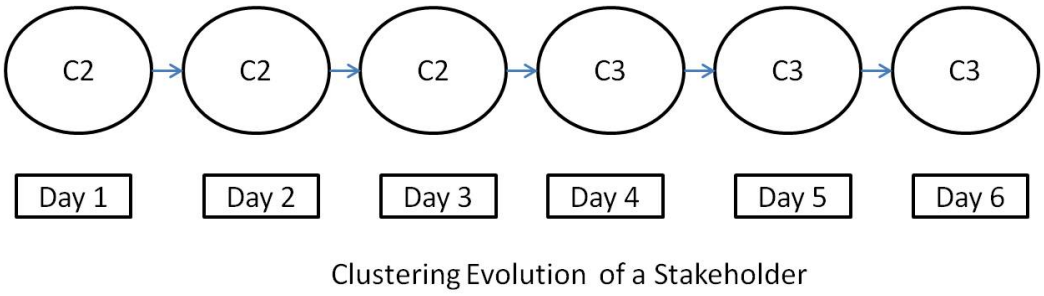

Figure 6.2. Clustering Evolution of a Stakeholder

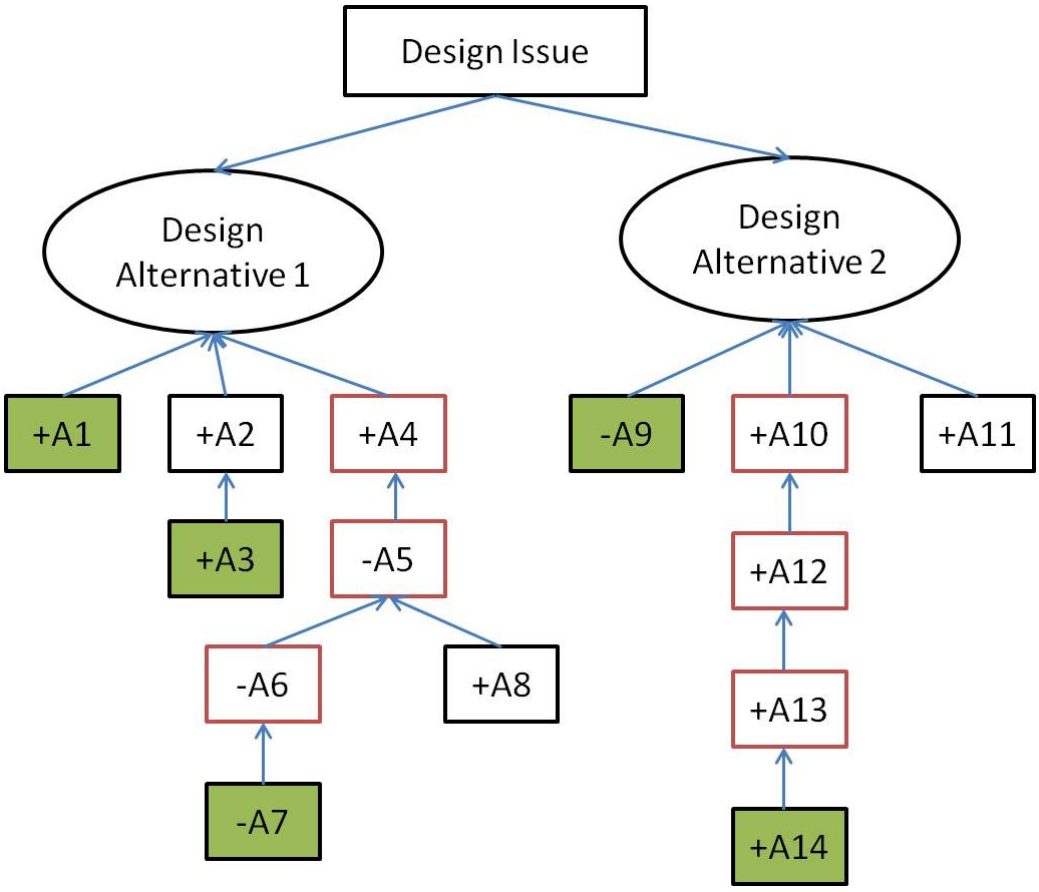

Figure 6.3. Sample Tree of a Stakeholder

ements marked in green like A1, A3, A7, A9 and A14 are the arguments posted by stakeholder S1. From the tree, we can see that arguments A1 and A3 supports design alternative 1 and argument A9 attacks design alternative 2. These arguments are posted by stakeholders during early stage of discussion and thus assigning himself to cluster group C2. But, A7 which is posted under A6 attacks design alternative 1 indirectly and similarly, A14 supports design alternative 2 based on argumentation reduction rules. From there on, the overall opinion weight of S1 on design alternative 1 has started decreasing to reach negative and weight on design alternative 2 has increased to reach positive and changing the stakeholder group to C3. The argument chains (A4, A5, A6) and (A10, A12, A13) are considered as influential argument chains that influenced S1 to change his opinion completely with the help of online collaborative discussion.

6.3. DETECTION OF ACTIVE AND INACTIVE STAKEHOLDERS

As said earlier that in an online decision making to resolve a design issue, stakeholders posts their viewpoints supporting or attacking a design alternative. Sometimes, some of the stakeholders actively participate in discussion by posting arguments responding to other arguments till the end of discussion. But, there are as well some stakeholders who participate in discussion initially and never contribute to discussion in later stages of decision making. We analyze the argumentation on each day and determine individual opinion weights of stakeholder on each design alternative on every day of discussion. An individual opinion on each design alternative for a stakeholder is computed by weighted summation of argument weights related to a stakeholder after argumentation reduction when all the arguments point directly to the design alternative. We analyze the individual opinions of stakeholder on each day of discussion and verify if any of the opinion weights of stakeholders change over time. If, the individual opinion of a stakeholder on any of design alternatives does not

change after initial stages of discussion, then it implies that he is inactive in the process and we classify such stakeholders as *Inactive Stakeholders*. On other hand, the opinion weights of certain stakeholders all through the end of discussion. Such stakeholders are classified as *Active Stakeholders* as they are continuously contributing to the discussion.

Consider the evolution of individual opinions of two stakeholders S1 and S2 for a period of 5 days as shown in Figure 6.4. Stakeholder's S1 individual opinions weights on the two design alternatives changes their values all through the discussion depicted by variation of colors. On other hand, stakeholder S2's opinions remains unchanged after Day 3 till end of discussion on Day 5. In this scenario, we classify S1 as *Active Stakeholder* whereas S2 is classified as *Inactive Stakeholder*.

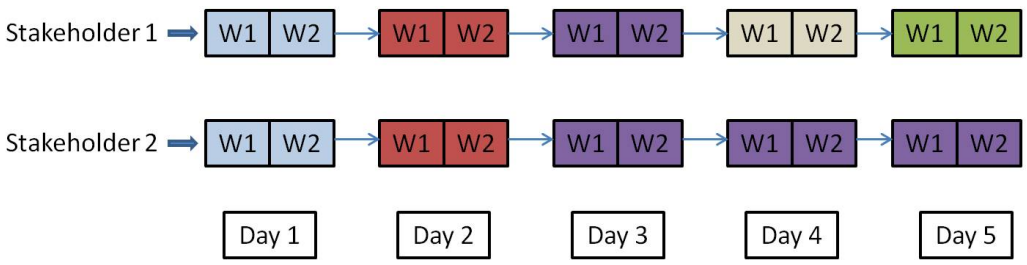

Figure 6.4. Evolution of Individual Opinions of a Stakeholder

6.4. DETECTION OF HOT ARGUMENTS

During argumentation, a stakeholder posts their arguments directly under a design alternative or in response to posts of other stakeholders. In this process, certain arguments gain a lot of attention and attract significant number of arguments following it. These arguments tend to increase the discussion by influencing various stakeholders to participate and effectively aid in decision making. Since, the argumentation process to resolve a design issue is spanned for several days, we analyze the discussion and detect viewpoints which have gained attention in process on each day

and also viewpoints that have significant number of follow up arguments from the start of the discussion. We classify these arguments or viewpoints as *Hot Arguments* in the discussion and detect such arguments on daily basis and overall basis since the start day of discussion. The *Hot Arguments* are determined based on number of follow up arguments an argument receive in discussion i.e., we compute the number of nodes from the current node till we reach the leaf node in all its branches in an argumentation tree.

Consider a sample argumentation tree showing a design alternative 1 for a design issue and various arguments posted for the alternative as shown in Figure 6.5. In order to compute the *Hot Arguments* based on this argumentation tree, we compute the number of follow up arguments each argument has received in the discussion process. For argument A1, there are totally 6 follow up arguments (A4, A5, A6, A7, A8, A9) which refer it either directly or indirectly. Similarly for argument A3, there is only one follow up argument A10. The count for A1 would be 6 whereas count for A3 would be 1.

Table 6.3 shows the follow up arguments for each argument in the tree. We can see that A1 has received most number of follow up arguments and has gained attention from significant number of stakeholders. Based on above analysis, we detect such arguments on each day during discussion process and also the arguments that are happening since the start of discussion till the day.

The number of follow up arguments for each argument is computed in the argumentation tree. Based on this computation, we determine the number of follow up arguments a stakeholder received for his posts. From the argumentation tree as shown in Figure 6.5, if arguments A1 and A3 are posted by a stakeholder S1, then we determine the total number of follow up arguments for a stakeholder by summation of arguments A1 and A3 received in discussion as in Equation 6.1. This helps us to understand the most happening stakeholder in the argumentation process i.e. we

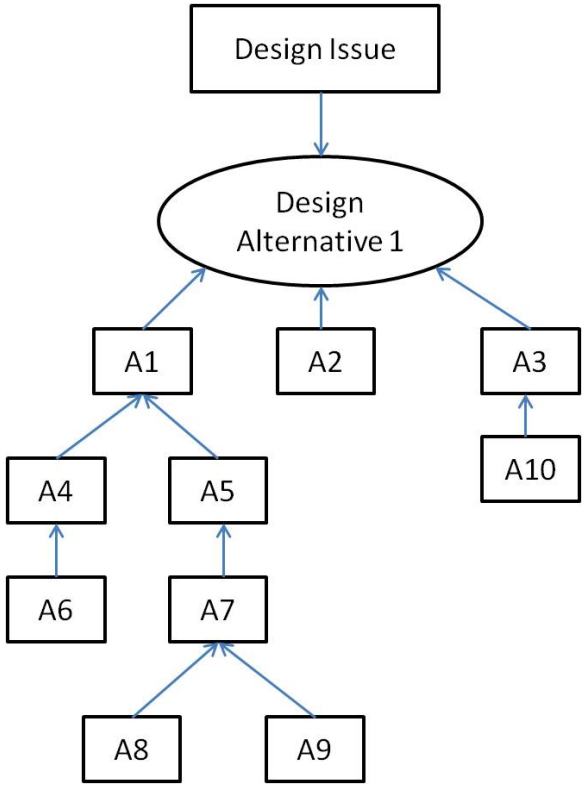

Figure 6.5. A Sample Argumentation Tree

Table 6.3. Argument ID vs Number of Follow up Arguments

Argument ID	Number of Follow up Arguments
A1	6 (A4, A5, A6, A7, A8, A9)
A2	0
A3	1 (A10)
A4	1 (A6)
A5	3 (A7, A8, A9)
A6	0
A7	2 (A8, A9))
A8	0
A9	0
A10	0

determine a stakeholder whose viewpoints have gained significant attention in decision making. We compute this again on daily basis on each day of argumentation.

$$Numberof FollowupArguments(S1) = FollowupArguments(A1)+ \\ FollowupArguments(A2) \quad (6.1)$$

6.5. DETECTION OF TRENDING TOPICS

During the argumentation process, stakeholders post their viewpoints in various perspectives. The arguments cover various topics of discussion and as the online discussion evolves, there is large number of viewpoints posted and its often difficult for stakeholders to understand the state of argumentation. The argumentation talks about various topics that affect the choice of a design alternative. It is a tedious task for a stakeholder to study each argumentation in the tree to figure out topics of interest. Some of the arguments might refer to similar topics and identification of such topics of interest enables the stakeholders to understand the topics that are most discussed and that have gained great attention in decision making and a major concern for stakeholder. The arguments that talk about similar topics needs to be detected by analyzing the argumentation tree and grouping such arguments makes us to understand the significance of the topic.

The entire argumentation tree has to be analyzed to detect such topics. We propose a use of text clustering technique that clusters the arguments based on their text similarity and identifies the topics of interest. We make use of Lingo algorithm [26] that is generally used to cluster web search results to perform text clustering of arguments posted by stakeholders. The Lingo algorithm outputs different clusters which are denoted by a cluster label. The cluster label depicts the topics that are discussed as part of the cluster group. This cluster label is identified as topic of interest of the arguments within that cluster. We perform the clustering process

on each day of the argumentation process to detect the topics of interest that are discussed till that day. In this way, the stakeholders will understand the state of argumentation till that day and also the topic that is discussed most often . As we perform the topic analysis on each day, we can also visualize the increase in demand for a particular topic as the discussion is evolved for several days.

7. EXPERIMENTAL EVALUATION

In this Section we describe in detail about the experimental setup and we showcase the results of various approaches discussed earlier. The results from the study indicate that our system is effective in capturing and supporting stakeholders in a collaborative decision making for software architecture design.

7.1. EXPERIMENTAL SETUP

In order to demonstrate the effectiveness of the proposed system a group of 50 students from Software Engineering class were recruited and presented with a hypothetical case study and were asked to resolve the design issue stated collaboratively using ISARCS. The students are considered as stakeholders to design the architecture of the system by resolving the design issue collaboratively. Apart from students as stakeholders, the case study is presented to three industrial experts who are having an experience of more than 10 years in software development. They were asked to choose a design alternative by reading the case study thoroughly. The result obtained from the experts is as well presented and compared with the result obtained by stakeholders in online discussion.

The stakeholders are presented with a case study where a hypothetical company is in process of modifying its existing product due to some change requirements. In order to modify the existing product, the company should initially analyze the requirement into lower level requirements and should accordingly modify its existing software architecture. In order to address the requirements, the company has came across a design issue that needs to be resolved and should capture a structured rationale along with maintaining linkage with the change requirement and the software architecture. The case study consists of 19 software requirements, 3 architectural

constraints and 25 architectural components and connectors.

The product that the company presently owes and wants to modify is called *Collab* that is used by various organizations or educational institutions to solve their internal organizational issues by effective collaborative discussion among their staff. While using the company's software users expressed their concerns about the lack of ability to draw and depict their ideas visually. This made the company to revisit their product and modify. Figure 7.1 shows the architecture for software *Collab*.

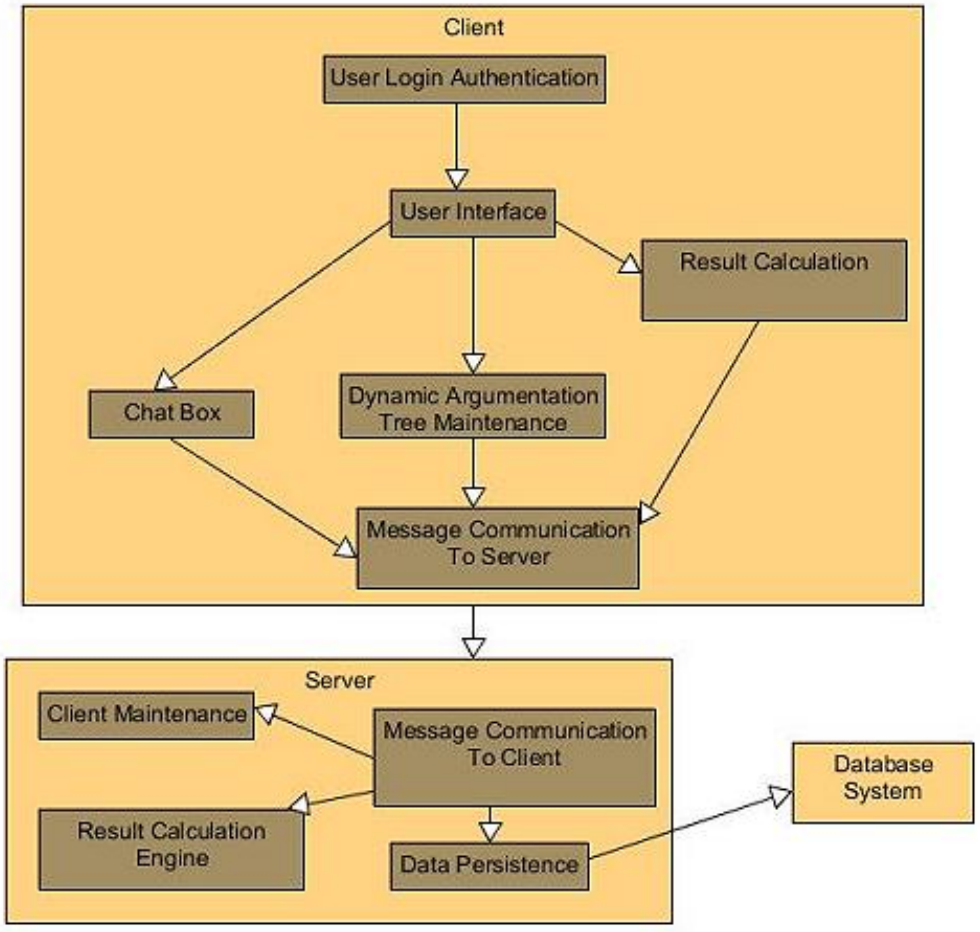

Figure 7.1. Software Architecture of Collab

The architecture design issue that arise from change requirement here is to

provide users a platform in order to draw and describe their ideas visually. The proposed design issue can be resolve in different ways using different design solutions. In order to resolve the design issue, three design alternatives are proposed of which one of the alternative needs to be selected as a design solution.

Design Alternative 1: Drawing Canvas Integrated in System that needs to be developed from scratch.

Design Alternative 2: Integrate a third party Software Drawing Tool into present system.

Design Alternative 3: Use of native paint application of Operating Systems and share the desktop with all users connected to system.

7.2. DATA COLLECTION

During the experiment, the stakeholders were provided with the required information regarding the case study and were given a week to understand it. The students were then asked to choose a best design alternative among available 3 design alternatives satisfying 19 software requirements stated. The experiment is carried out for a period of 2 weeks where the students actively participated in discussion contributing 413 arguments, 42 evidences and 628 architectural relations. In order to ensure that the arguments posted are relevant and related to context of discussion, the entire discussion is moderated by a moderator. The main goal of the experiment is to resolve a design issue for software architecture design collaboratively using ISARCS and to capture the argumentation data that can be used for further data analysis to provide more depth view of entire online discussion. The case study is also shared to three industrial experts and the experts were given a period of 2 weeks to understand the case study. After the experts have studied the case study, they have responded with their own design decision and respective rationales individually.

7.2.1. ISARCS Discussion. A class of students participated in an online discussion to resolve a design issue that is presented as a case study for a period of two weeks. Each of the students is provided with credentials that will enable him to login and post his arguments by appropriately selecting the related requirements and architectural elements. There were totally 413 arguments, 42 evidences and 628 architectural relations captured during online discussion.

The stakeholders participated in online discussion using ISARCS explaining their views in support or attack of a design alternative. Figure7.2 shows sample argumentation tree of design issue discussion using ISARCS where the design issue, design alternatives and arguments are linked to software requirements and architecture elements. There are numerous requirements during software architecture design. Only certain requirements are related to design issue under discussion. Table 7.1 shows sample requirements list that are related to design issue stated as part of case study.

Table 7.1. Sample Requirements List

Requirement ID	Requirement Description
R1	Users may be given different drawing options and pre defined drawing patterns (like circle or rectangle) that enables them to draw and visualize their ideas quickly.
R2	The change requirement should be implemented, tested and deployed in short period of time.
R3	When a user edits a diagram, his changes need to be broadcasted to all users who are connected to the system efficiently.
R4	The entire system should be easily maintainable
R5	Multiple Users connected to the system must be able to work collaboratively on the same diagram simultaneously.

As pointed earlier, stakeholders have to explicitly mention the weight of support or attack of their arguments. Table 7.2 shows sample arguments list posted by the stakeholders for one of the design alternatives. Using the argument weights, we use fuzzy argumentation reduction inference engine to compute the design alternative

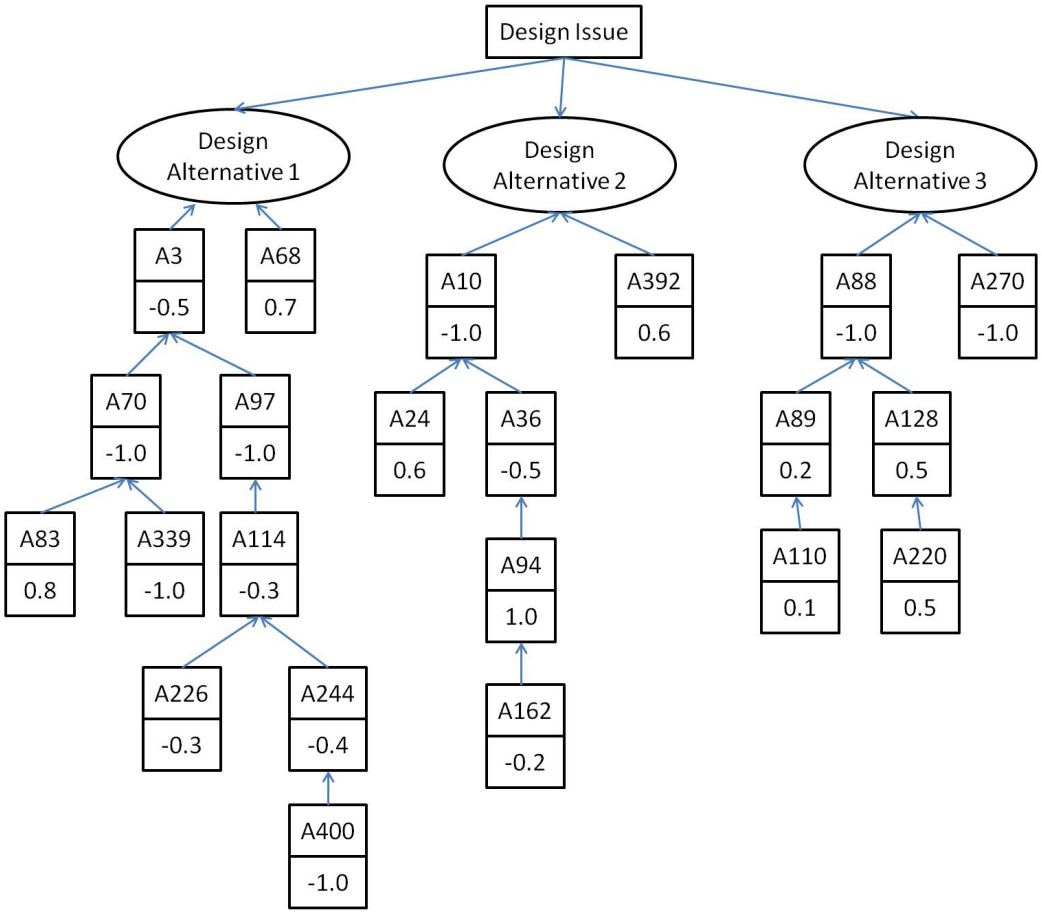

Figure 7.2. Sample Argument Tree from Experiment

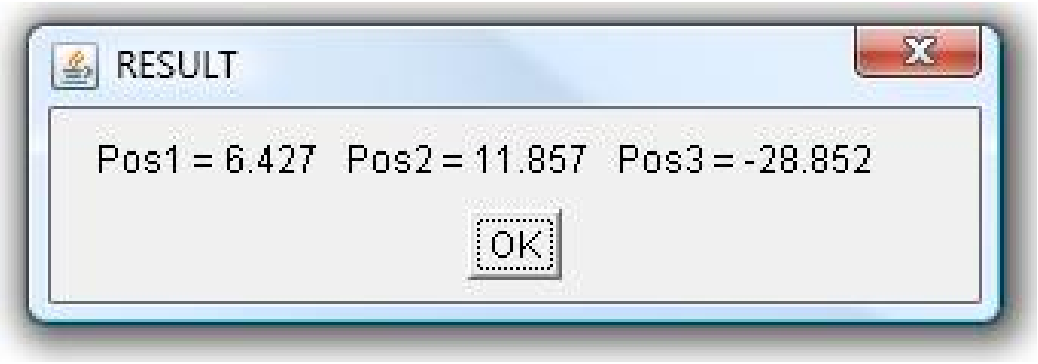

Figure 7.3. Design Decision using ISARCS

that is favored by most of the stakeholders during discussion. As per result computed by ISARCS, the design alternative 2 is supported by most of the stakeholders while design alternative 3 is attacked by majority of them as shown in Figure 7.3. This result helps architects to choose a design alternative and design the architecture for system.

Table 7.2. Sample Argument List on Design Alternative 1

Argument ID	Argument Description	Strength of Argument
A4	The cost of paying someone to sit down and write out this drawing tool would be very high and would also take away from development time elsewhere.	-0.5
A13	If you are to make something from scratch you can not know if it is user friendly without user testing which takes a lot of time. The other two options have been tested by a lot of users and have ratings online already for you to see.	-0.5
A68	Also, using a third party software creates the risk that an updated version may remove components used for designing their ideas. This risk would not be present in a Position 1 because the system would be self-designed.	0.7
A19	If we design this system from scratch then the responsibility to support it, add new features, and fix bugs lies entirely on us. I think it would be wiser to lay that responsibility off on some other organization.	-0.5
A74	Agreed, while the other options are cheaper, and faster methods, there is definitely the positive of easily adding new functionality if it is needed in the scratch-made tool.	0.3

7.2.2. Industrial Experts Opinion. As said earlier that the case study is also presented to three industrial experts. The experts had gone through the entire case study and provided their design decision. Out of the three experts, two of the experts presented their decision as design alternative 2 and one of them as design alternative 1. The majority of the experts shared their decision as design alternative 2. When we compare the result obtained from ISARCS with that of experts decision, we can see that the results are similar. The collaborative discussion of inexperienced professionals using ISARCS has yielded a design decision which is incongruence to the decision of industrial experts. This proves the effectiveness of the system in its support to software architecture design using collective decision making process.

7.3. INTELLIGENT DATA ANALYSIS

7.3.1. Analysis of Software Concerns. Huge argumentation tree that is captured during discussion is further analyzed to provide stakeholders with overall view of effect of concerns on each design alternative. The requirements list is studied manually and a list of concerns are shortlisted. Based on these concerns list, the entire argumentation is parsed to compute the effect of each concern on three design alternatives using Algorithm 1. Table 7.3 shows the result.

Table 7.3. Effect of Concerns on Design Alternatives

Software Concern	Design Alternative 1	Design Alternative 2	Design Alternative 3
Cost	2.766	5.337	-5.1
Testing Effort	5.828	6.928	-0.2139
Development Time	-4.064	5.423	-6.1826
Development Effort	0.1	1.0	-1.048
Software Maintenance	6.3640	2.04	-3.2
Security	1.71	3.0	-3.375
Licensing Issue	0.63	-0.572	-0.5
Architecture Complexity	0.5	-0.8	-3.339

The results of Algorithm 1 help the stakeholders to understand how each concern would affect their choice of design alternative. As per results, the design alternative 1 is supported more for software maintenance concern but the same alternative is attacked by most people in case of other concerns like cost and testing effort. With the help of these results, stakeholders now tradeoff among these concerns and choose an alternative that best suits their project needs and priority.

7.3.2. Classification of Stakeholders Viewpoints. The stakeholders posted arguments supporting or attacking various design alternatives all through the argumentation experiment. The data that is captured after entire discussion is analyzed to compute the collective opinions of a group on individual arguments. As part of this discussion, since there were totally 413 arguments posted, the collective opinion of a group on all the arguments is computed to determine the arguments that are most favored or attacked through the discussion.

Table 7.4 shows the collective opinion values for some of the arguments from the argumentation tree. The arguments listed in the table have high follow up arguments and high collective determination values when compared to other arguments in the tree. From the table, we can see that argument A7 is having highest collective determination value when compared to all other arguments and is collectively supported by posting 6 follow up arguments. Since A7 is the argument having highest collective determination value, we classify the argument as *Decision Centric Argument*. This argument A7 totally has a positive impact on the design alternative. Similarly, argument A6 has the least collective determination value and also is classified as *Decision Centric Argument*. The argument A6 has a significant negative impact on the design alternative and understanding of such arguments helps the stakeholders in the decision making process. On other hand, when we consider arguments A1 and A12, though the collective determination value of these arguments are trivial, they

have attracted more number of arguments into discussion when compared to all other arguments having significant values of collective determination. We classify these arguments A1 and A12 as *Attention Seeking Arguments* as they attracted most of the stakeholders to participate in discussion accumulating 16 and 20 follow up arguments respectively.

Table 7.4. Collective Opinion Analysis

Argument ID	Collective Support	Collective Attack	Collective Determination	Follow up Arguments
7	3.5	0	3.5	6
84	3.75	-2	1.75	10
4	3.67	-2.98	0.68	11
3	5	-4.58	0.41	14
1	5.3	-4.88	0.42	16
6	0.5	-8.1	-7.6	15
12	4.28	-9.22	-4.93	20
5	0.3	-6	-5.7	10
10	2.3	-4.5	-2.2	14
146	0.7	-2.18	-1.48	5

When we consider arguments having maximum collective support value, from the table we can see that argument A1 is having maximum collective support value suggesting that the argument is supported by most of the stakeholders. The argument A1 is classified as *Most Supported Argument* when compared to all other arguments during online discussion. Such argument helps stakeholders to understand the concern or context that most of the stakeholders agreed upon to resolve the design issue. On other hand, argument A12 has least collective attack value. With negative collective attack value, the argument illustrates that it has conflicting interest among stakeholders and is classified as *Most Attacked Argument*. Table 7.5 lists the arguments classified.

Table 7.5. List of Classified Arguments

Argument ID	Argument Description	Design Alternative	Argument Classification
7	If you are using a design tool native to the operating system then you most likely would not be able to have multiple users editing the same diagram without some extra code that you must write.	Design Alternative 3	Decision Centric Argument
6	In terms of cost I think this option would be the best.	Design Alternative 3	Decision Centric Argument
1	I support this position because there are enough drawing tools out there that can meet the necessary requirements and still be easy to integrate into the system.	Design Alternative 2	Attention Seeking Argument and Most Supported Argument
12	When using a 3rd party software you can only hope that it has been tested fully.	Design Alternative 2	Attention Seeking Argument and Most Attacked Argument.

7.3.3. Requirements Traceability. As stated earlier that every design issue is mapped to subset of software requirements that are related to the issue. In this case study, the proposed design issue is related to 19 software requirements. The design issue in the case study is related to introducing a new component into the software architecture to address the concern raised by users of the product. The design solution that is chosen after entire argumentation will result in addition of new component and connectors to existing components in the software architecture. After the argumentation reduction and weighted summation of arguments, design alternative 2 is the alternative that is favored by most of the stakeholders. The result computed by argumentation engine supports stakeholders in choosing alternative 2 as their design

solution. As per the decision, a new component Third Party Drawing Tool is added into the architecture design. This new component communicates to other component, Message Communication to Server through a connector. The resolution of the design issue resulted in two new architectural elements into software architecture. These two architectural elements are now related to the 19 software requirements mapped to the design issue as shown in Figure 7.4. Any modification to any of those requirements affects these two architectural elements and vice versa. In the requirements traceability matrix, the cells corresponding to these requirements and architectural elements are updated to 1 showing their dependency.

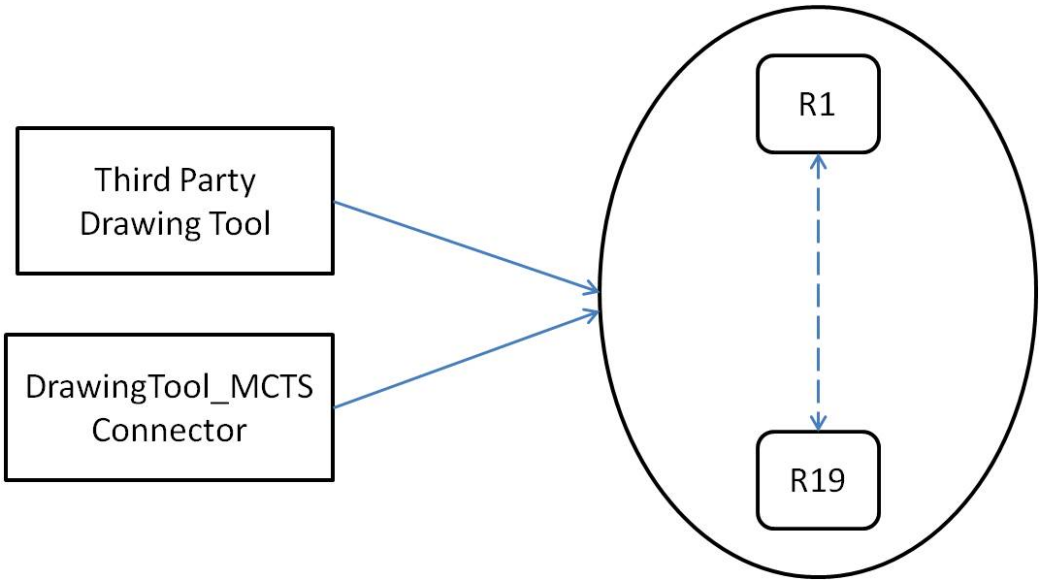

Figure 7.4. Traceability of Requirements and Architectural Elements

7.3.4. Topic Analysis. The entire argumentation data captured during online discussion is clustered using Lingo algorithm to detect the topics that are under discussion. Table 7.6 shows some of the topics that are discussed as part of issue resolution. The topics that are often discussed in issue resolution are Development Time and Cost. By this analysis, we can say that most of the stakeholders are concerned with those aspects of software development. On other hand, there are not

many arguments related to resource usage suggesting that the stakeholders are not much concerned with utilization aspect.

Table 7.6. Topic Analysis

Topic	Number of Arguments
Development Time	60
Testing	55
Open Source	22
Operating Systems	18
Cost	74
Maintenance	20
Security	57
Resource Usage	2

7.4. EVOLUTIONARY DATA ANALYSIS

7.4.1. Convergence Using Polarization Groups. As the experiment is conducted for a period of 2 weeks, the favorability of stakeholders participating in the discussion is computed for each day on individual alternatives based on their opinion weights. The stakeholders are clustered into different polarization groups. The clustering process is performed on each day of online discussion.

From Table 7.7, we can see the evolution of clusters during online discussion where the number of stakeholders in a cluster group change as the discussion is spanned for several days. There are stakeholders who are joining into the discussion on each day. Consider cluster group C6. It started with 1 stakeholder in its group after first day of discussion. But, as the discussion is evolved and as more and more participated in discussion, the number of stakeholders in the group gradually increased and reached 17 after final day of discussion. This cluster group C6 is the one which has more number of stakeholders when compared to other clusters. The stakeholders in this group support design alternative 2 and attack other two design alternatives.

Table 7.7. Clustering Evolution

Dates	C1	C2	C3	C4	C5	C6	C7	C8	Total Stake-holders
November 11th, 2014	1	2	0	0	0	1	1	0	5
November 12th, 2014	0	0	1	1	0	3	0	1	6
November 13th, 2014	0	0	2	1	0	3	0	1	7
November 14th, 2014	0	0	2	1	0	4	0	1	8
November 15th, 2014	0	0	4	1	0	4	0	1	10
November 16th, 2014	0	2	3	0	0	7	0	3	15
November 17th, 2014	0	2	7	0	1	8	1	4	23
November 18th, 2014	1	3	10	0	1	10	1	10	36
November 19th, 2014	8	8	7	3	3	12	1	2	44
November 22nd, 2014	3	4	11	1	1	17	2	6	45
November 23rd, 2014	3	4	12	1	1	17	2	6	46

The clustering evolution illustrates the point of convergence in discussion. The design alternative 2 would be most preferred design solution as most of the stakeholders agree with it.

7.4.2. Detection of Influential Argument Chains. During discussion, the stakeholders also change their opinion and move from one cluster to another cluster. The reason for stakeholders transition is the influence of certain argument chains in the discussion that have made them to change their opinion. Without loss of generality, we consider userids of two stakeholders as Stakeholder 1 and Stakeholder 2, who participated in the discussion and changed their opinion.

Table 7.8. Influential Argument Chain for Stakeholder 2 on Design Alternative 1

Argument ID	Argument Description	Argument Weight
A343(By Stakeholder 2)	*However, using a third party application would require us first to obtain their permission for use in our software, possbily permission to edit it if the tool doesnt already meet all the requirements perfectly, and we would need their support for any future changes that are made either to their tool or ours. The additional correspondence needed may make that option take more time than developing the tool from scratch, depending on how we rank in the third partys priorities*	*-0.4*
A321	I agree this would give the development team the greatest amount of control but according to requirement 145 the change needs to be made in a short period of time. Trying to create a new functionality quickly could lead to a lot more mistakes. Rushing through the testing process leads to a worse output then using an already working third party application.	-0.5
A269	Since it is an integrated Canvas, development team has the full control. That the entire system along with the new functionality should be tested thoroughly end to end is highly achievable. Therefore, I support position 1 on this.	1.0

Table 7.9 shows the transition of cluster groups for two stakeholders. Stakeholder 1 was part of cluster C8 on November 18th, 2014 where he agrees with neither of the alternatives. But, on November 19th, 2014, he moved to cluster group C6 supporting design alternative 2 and attacking other alternatives. Similarly, Stakeholder 2 changed his opinion and moved from cluster group C8 to C3 supporting design alternative 1.

Based on these transitions, we analyze the argumentation tree and detect the argumentations chains that have influenced these individuals to change their opinion and opt a different idea. Tables 7.10 7.8 shows the argument chains that have influ-

enced stakeholder 1 and stakeholder 2 to change their decisions respectively.

Table 7.9. Transition of Cluster Groups of Two Stakeholders

Stakeholder ID	November 18th, 2014	November 19th 2014
Stakeholder 1	C8 *(Negative, Negative, Negative)*	C6 *(Negative, Positive, Negative)*
Stakeholder 2	C8 *(Negative, Negative, Negative)*	C3 *(Positive, Negative, Negative)*

Table 7.10. Influential Argument Chain for Stakeholder 1 on Design Alternative 2

Argument ID	Argument Description	Argument Weight
A320 (By Stakeholder 1)	*I agree, with a limited amount of time this is the best option to have an application that works. Time can be spent testing the third-party application instead of trying to create one from scratch.*	*1.0*
A194	This position will have the fewest amount of bugs, because the third party tool has been in use and tested for a while, and there is an entire community of developers committed to improving it. While I cannot say it will be defect-free in our implementation, a lot of the testing and improvement has already been done. This also takes out the time of developing the full tool, and requirement 145 says it should happen in a short period of time.	1.0

7.4.3. Detection of Hot Arguments. As the online discussion evolves on each day of argumentation, we compute the number of follow up arguments for each argument posted during discussion. Based on the number of follow up arguments, we detect the arguments that have gained significant attention in the argumentation. The system computes the Hot Arguments on each day and as well the Hot Arguments all through the discussion till that date. Table 7.11 shows the *Hot Arguments* till the date from the beginning of discussion. A12 is the argument that has gained more

number of follow up arguments when compared to others at the end of discussion. Similarly, Table 7.12 shows the *Hot Arguments* on each day. We can see that, on each day different arguments stand out and receive significant number of follow up arguments in discussion. A12 is the arguments which have received large number of arguments on a single day.

Table 7.11. List of Hot Arguments Since Start of Discussion (Overall Basis)

Date	Argument ID	Number of Follow up Arguments
November 11th, 2014	A7	2
November 12th, 2014	A1, A2	3
November 13th, 2014	A1,A2	4
November 14th, 2014	A1,A2,A11	4
November 15th, 2014	A11	5
November 16th, 2014	A31	6
November 17th, 2014	A10	10
November 18th, 2014	A12	13
November 19th, 2014	A12	20
November 22nd, 2014	A12	20
November 23rd, 2014	A12	20

Table 7.12. List of Hot Arguments on Each Day (Daily Basis)

Date	Argument ID	Number of Follow up Arguments
November 11th, 2014	A7	2
November 12th, 2014	A1,A2,A10,A11,A14,A31	2
November 14th, 2014	A11	2
November 16th, 2014	A3	3
November 17th, 2014	A10, A6	5
November 18th, 2014	A12, A3	6
November 19th, 2014	A12	7
November 23rd, 2014	A1, A3, A201	2

The number of follow up arguments received for each stakeholder's arguments is also computed and the stakeholders who have received most number of follow up arguments has increased the participation in the online discussion. Table 7.13 shows a sample list of stakeholders and the number of follow up arguments they have received.

Table 7.13. Number of Follow up Arguments Received by Stakeholders

Stakeholder	Number of Follow up Arguments
Stakeholder 1	106
Stakeholder 2	48
Stakeholder 3	36
Stakeholder 4	26

Stakeholder 1 has received significant number of follow up arguments in discussion.

7.4.4. Detection of Active and Inactive Stakeholders. After entire argumentation, the opinion weights on various design alternatives for each of the stakeholders in computed on daily basis. Some of the stakeholders have consistently participated in the discussion, whereas some of the stakeholders participated only during initial days. Table 7.14 shows the opinion weights on design alternatives for one of the stakeholder who participated in discussion for each day. The opinion weights from the table depicts that the stakeholder was active in discussion only for the first 3 days of discussion and for the next 9 days of discussion, he has not posted a single argument. Such stakeholders are classified as *Inactive Stakeholders.*

Table 7.14. Evolution of Opinion Weights on Design Alternatives for a Stakeholder

Design Alternative/Dates	November 11th, 2014	November 12th, 2014 to November 23rd, 2014
Design Alternative 1	1	2.3
Design Alternative 2	0	-2.125
Design Alternative 3	0	1.625

7.4.5. Detection of Trending Topics. On each day of argumentation, the argumentation data that is captured is analyzed to detect the topics that are most discussed till that day. The arguments are clustered using Lingo algorithm based on the topic they discuss. Table 7.15 shows the evolution of topics for some of

Table 7.15. Evolution of Trending Topics

Dates/Topics	Testing	Cost	Development Time	Security	Operating Systems
November 11th, 2014	4	4	2	0	0
November 12th, 2014	5	6	2	5	0
November 13th, 2014	6	7	2	5	0
November 14th, 2014	9	8	2	7	2
November 15th, 2014	10	8	2	8	2
November 16th, 2014	20	14	2	15	7
November 23rd, 2014	55	74	60	57	18

the days of argumentation. As part of the discussion, we can see that most of the stakeholders discuss a lot about Testing, Cost, Development Time and Security. Some of the stakeholders are also concerned with the dependency of operating system during product development.

The table shows the most discussed topics during argumentation. There are other topics under discussion as well, but they haven't grasped a lot of attention from the group.

8. CONCLUSION AND FUTURE WORK

In this thesis, we presented and discussed about ISARCS that enables architects to capture and maintain the software architecture rationale for collaborative architecture design. The entire architectural rationale maintains its links to software requirements and architectural elements that aids stakeholders to understand the relationships between various elements during software development. We also presented various techniques to analyze the architectural rationale in different perspectives and support stakeholders to understand the state of argumentation that aids them in choosing a well suited design solution. Finally, we conducted a case study and provided with the results that indicate the effectiveness of the proposed system and intelligent analysis of architecture rationale.

In future, if multiple products encounter similar 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. The architectural knowledge captured can be reused across products to resolve similar design issues. Apart from that, the architectural rationale captured can also be analyzed with advanced data mining techniques to understand more about argumentation.

BIBLIOGRAPHY

[1] JanSalvador van der Ven, AntonG.J. Jansen, JosA.G. Nijhuis, and Jan Bosch. Design decisions: The bridge between rationale and architecture. In AllenH. Dutoit, Raymond McCall, Ivan Mistrk, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 329–348. Springer Berlin Heidelberg, 2006.

[2] Dewayne E Perry and Alexander L Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[3] Anton Jansen and Jan Bosch. Software architecture as a set of architectural design decisions. In *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*, pages 109–120. IEEE, 2005.

[4] Rafael Capilla, Francisco Nava, Sandra Pérez, and Juan C Dueñas. A web-based tool for managing architectural design decisions. *ACM SIGSOFT software engineering notes*, 31(5):4, 2006.

[5] Cristina Gacek, Ahmed Abd-Allah, Bradford Clark, and Barry Boehm. On the definition of software system architecture. In *Proceedings of the First International Workshop on Architectures for Software Systems*, pages 85–94. Seattle, Wa, 1995.

[6] Fabian Gilson and Vincent Englebert. Rationale, decisions and alternatives traceability for architecture design. In *Proceedings of the 5th European Conference on Software Architecture: Companion Volume*, page 4. ACM, 2011.

[7] Meiru Che and Dewayne E. Perry. Exploring architectural design decision management paradigms for global software development. In *The 25th International Conference on Software Engineering and Knowledge Engineering, Boston, MA, USA, June 27-29, 2013.*, pages 8–13, 2013.

[8] Rafael Capilla, Francisco Nava, Jesús Montes, and Carlos Carrillo. Addss: Architecture design decision support system tool. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 487–488. IEEE Computer Society, 2008.

[9] Muhammad Ali Babar and Ian Gorton. A tool for managing software architecture knowledge. In *Proceedings of the Second Workshop on SHAring and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent*, SHARK-ADI '07, pages 11–, Washington, DC, USA, 2007. IEEE Computer Society.

[10] Xiaofeng Cui, Yanchun Sun, and Hong Mei. Towards automated solution synthesis and rationale capture in decision-centric architecture design. In *Software*

Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on, pages 221–230. IEEE, 2008.

[11] I. Lytra, Huy Tran, and U. Zdun. Constraint-based consistency checking between design decisions and component models for supporting software architecture evolution. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 287–296, March 2012.

[12] J. Savolainen and T. Mannisto. Conflict-centric software architectural views: Exposing trade-offs in quality requirements. *Software, IEEE*, 27(6):33–37, Nov 2010.

[13] Lars Bratthall, Enrico Johansson, and Björn Regnell. Is a design rationale vital when predicting change impact? a controlled experiment on software architecture evolution. In *Proceedings of the Second International Conference on Product Focused Software Process Improvement*, PROFES '00, pages 126–139, London, UK, UK, 2000. Springer-Verlag.

[14] Stephen Toulmin. The uses of argument. 1958. *Cambridge: Cambridge UP*, 2003.

[15] Jeff Conklin and Michael L. Begeman. gibis: A hypertext tool for exploratory policy discussion. *ACM Trans. Inf. Syst.*, 6(4):303–331, October 1988.

[16] N Papadias. Hermes: Supporting argumentative discourse in multi agent decision making. In *Proceedings of the 15th National Conference on Artifical Intelligence (AAAI-98)*, pages 827–832, 1998.

[17] Chenn-Jung Huang, Yu-Wu Wang, Tz-Hau Huang, Jia-Jian Liao, Chun-Hua Chen, Chuan-Hsiang Weng, Yu-Jen Chu, Chiao-Yun Chien, and Hung-Yen Shen. Implementation and performance evaluation of an intelligent online argumentation assessment system. In *Electrical and Control Engineering (ICECE), 2010 International Conference on*, pages 2560–2563. IEEE, 2010.

[18] Xiaoqing Liu, NagaPrashanth Chanda, and Eric Barnes. Software architecture rationale capture through intelligent argumentation. In *Proc. of 2014 International Conference on Software Engineering & Knowledge Engineering*, Vancouver, Canada, July 2014.

[19] R.S. Arvapally and Xiaoqing Liu. Collective assessment of arguments in an online intelligent argumentation system for collaborative decision support. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages 411–418, May 2013.

[20] Xiaoqing Frank Liu, Maithili Satyavolu, and Ming C Leu. Contribution based priority assessment in a web-based intelligent argumentation network for collaborative software development. In *Collaborative Technologies and Systems, 2009. CTS'09. International Symposium on*, pages 147–154. IEEE, 2009.

[21] Xiaoqing(Frank) Liu, Samir Raorane, and MingC. Leu. A web-based intelligent collaborative system for engineering design. In W.D. Li, Chris McMahon, S.K. Ong, and AndrewY.C. Nee, editors, *Collaborative Product Design and Manufacturing Methodologies and Applications*, Springer Series in Advanced Manufacturing, pages 37–58. Springer London, 2007.

[22] Xiaoqing Frank Liu, Ekta Khudkhudia, Lei Wen, Vamshi Sajja, and M Leu. An intelligent computational argumentation system for supporting collaborative software development decision making. *Artificial Intelligence Applications for Improved Software Engineering Development, Farid Meziane and Sunil Vadera, Hershey, PA: IGI Global*, pages 167–180, 2009.

[23] Scott Sigman and Xiaoqing Frank Liu. A computational argumentation methodology for capturing and analyzing design rationale arising from multiple perspectives. *Information and Software Technology*, 45(3):113 – 122, 2003.

[24] Philippe Kruchten, Patricia Lago, and Hans van Vliet. Building up and reasoning about architectural knowledge. In Christine Hofmeister, Ivica Crnkovic, and Ralf Reussner, editors, *Quality of Software Architectures*, volume 4214 of *Lecture Notes in Computer Science*, pages 43–58. Springer Berlin Heidelberg, 2006.

[25] Eric Barnes and Xiaoqing Liu. Text-based clustering and analysis of intelligent argumentation data. In *Proc. of 2014 International Conference on Software Engineering & Knowledge Engineering*, Vancouver, Canada, July 2014.

[26] Stanislaw Osinski and Dawid Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54, May 2005.

[27] Bibb Latan and Sharon Wolf. The social impact of majorities and minorities. *Psychological Review*, 88(5):438–453, Sep 1981.

[28] Bibb Latan. The psychology of social impact. *American Psychologist*, 36(4):343–356, Apr 1981.

VITA

NagaPrashanth Chanda earned his Bachelors degree in Electrical and Electronics Engineering from Osmania University, India in 2010. After completion of his bachelors, he worked as a Senior Systems Engineer at Infosys Limited, Hyderabad, India for 3 years (till July 2013). He has been a graduate student in the Computer Science Department at Missouri University of Science and Technology since August 2013 and worked as a Graduate Research assistant under Dr. Xiaoqing (Frank) Liu from August 2013 to till date. He received his Masters in Computer Science at Missouri University of Science and Technology in May 2015.