IEMS Vol. 7, No. 3, pp. 266-287, December 2008.

An Argument-based Approach to Manage Collaborative Negotiations in Software Systems Design

Stephen C-Y. Lu
The IMPACT Research Laboratory Viterbi School of Engineering
University of Southern California Los Angeles, California 90089 USA
Tel: +1-213-740-9616, E-mail: sclu@usc.edu

Nan Jing†
The IMPACT Research Laboratory Viterbi School of Engineering
University of Southern California Los Angeles, California 90089 USA
Tel: +1-213-740-9616, E-mail: jing@usc.edu

Selected paper from APIEM 2006

Abstract. To manage collaborative negotiation in software system design, we have built a socio-technical argument-based negotiation management approach by integrating a Socio-technical Co-construction Process (STCP) with an Argument-based Negotiation Process (ABNP). This paper reviews relevant research work and presents each step of this approach. The STCP provides rich contextual information of technical decisions and social interactions in a system design process. The ABNP provides STCP with a negotiation management and conflict resolution strategy by guiding software engineers to generate, exchange and evaluate their argument claims in negotiation activities. In addition, this paper describes a prototype system which implements this new approach using the advanced Web-based software technologies with the goal of demonstrating how to systematically enhance the negotiation management capabilities in a dynamic socio-technical framework.

Keywords: Negotiation, Collaboration, Design, Software, Argument, Process Management

1. INTRODUCTION

Software engineering aims to support the engineering team to reach technical agreement and make rational decisions for software design tasks. It creates technical solutions to information processing problems through the use of scientific methods. Software engineering research is concerned with how to improve the quality and efficiency of software design decisions in order to predict, analyze, implement and maintain software systems that can satisfy complex and evolving customer requirements. One unique characteristic of software engineering is that "software is design-intensive, as manufacturing (such as the repeated production of program codes) cost is a relatively minor component of software production cost" (Aldrich *et al*., 2006). Nowadays driven by market globalization, outsourcing and Internet revolution, most software design is carried out by distributed software teams that include developers, architects and managers who have varying backgrounds and expertise. When people from different disciplines work collaboratively over the distance and time boundary, the software design process becomes very complicated since such collaborative design endeavor involves numerous technical and social (or socio-technical) issues. In this highly distributed and connected global software industry, engineers have to count on good collaboration methodologies to stay competitive when designing, developing and deploying complex software systems. Compared with traditional individual design, collaborative design has some intrinsic characteristics. The objectives of design are not homogeneous since various persons join the software design team and make decisions collaboratively based on their own objectives. A better understanding of how software designers should collaborate with each other in technical tasks under social interactions to make rational group decisions is the challenge, which needs to be confronted by the whole engineering research community. However, despite its importance, the current investigation of the features and characteristics of collaboration software design is more limited to practiced heuristics rather than scientific principles with solid theoretical foundation.

In real-life software design processes, software engineers always need to negotiate with each other in order to

† : Corresponding Author

reach agreements based on collective rationality when they have conflicting opinions and competing demands. The ability to negotiate with multiple stakeholders who have different technical expertise and diverse social backgrounds (e.g., many other non-technical factors) is just as important as the ability to develop computation algorithms and build data structures in software design. This ability is the key to make a rational group decision in light of different objectives, criteria and preferences. In a software design team involving people with various backgrounds and expertise, each individual can only deal with a portion of the overall problem. Since the pace and scope of knowledge continue to expand, the amount of information communicated increases significantly during the collaboration process. As collaborative design usually deals with large and complex software systems, designers have to collect and interpret various information by dealing with different data structure and algorithms. Therefore, at least two significant challenges occur. One is to efficiently manage negotiation activities for collective rationality in software design process. The other is the identification and resolution of the conflicts generated when different individuals share the information (Wall *et al*., 1995). If poorly supported and managed, both can cause the delays in progress, the wastes of resources, or even the failure of the whole project. In order to effectively achieve collective rationality in software design tasks, these two challenges must be addressed with sound theories and good technologies.

In traditional software engineering practices, little attention was placed on these collaborative activities of software design, let alone systematic supports to negotiation tasks and collective rationality. Typical software engineering methodologies do not explicitly address the resolution of conflict, either. These methodologies are more geared towards maintaining consistency and do not allow conflicts to be systematically expressed, not to mention being constructively resolved. Indeed, existing software methodologies could be characterized as conflict avoidance, in that they prescribe particular approaches to assist software practitioners in breaking problems down and resolving conflicting issues in particular ways by each individual stakeholder. While this approach helps to avoid conflicts during development, it does not help to revolve and harvest conflicting opinions in collaborative design. Most existing methodologies require a single consistent software design as a basis for a coherent system solution, which means that conflicts are suppressed when the design is plotted. The ability to detect the presence of conflict and systematically resolve conflicts in more explicit manners (such as negotiation) is a significant step forward in software design methodology.

Collaborative negotiation and conflict resolution have been well-researched areas for the past several decades. Although they have been examined from several perspectives including social science, economics, decision theory, engineering design, and artificial intelligence (e.g., Nash 1950, Kenney and Raiffa 1976, Raiffa 1982, Davis and Smith 1983, Sycara 1991, Bui 1987, Shakun 1992), it is only recently that frameworks for managing negotiation and providing decision support for conflict resolution have emerged (e.g., Kersten 1985, Durfee and Lesser .L989, Hunhs and Gasser 1989, Jelassi and Foroughi 1989, Kersten *et al*., 1991, Sycara 1989, Lim and Benbasat 1991, Bui 1993, Bui 1994). In order to develop a sound negotiation framework, it is essential to distinguish between the negotiation content, i.e. the substantive issues which are to be discoursed and resolved, and the negotiation process, i.e. the procedure of dealing with substantive issues or the sequence of events leading to the final result of the negotiation. In practice, the process via which the parties negotiate their conflicts determines the results, the quality and the efficiency of a negotiation at least as much as the substance which they are negotiating about. Dealing simultaneously with both substance ("what") and process ("how") is a challenge in negotiation research. Very often, negotiators tend to focus most of their attention on the substance, thereby neglecting to take care of the process. The challenge that our research tries to address is to help the software design teams to prepare and follow a well-managed negotiation process seamlessly and systematically in order for them to deal with the substance, i.e. software design solution, in the best possible and most productive way.

Specifically, this paper presents an argument-based approach to manage and support negotiation process in collaborative software design by modeling, tracking and managing stakeholders' negotiation arguments, which continuously evolve during the collaborative process of software design. This approach helps stakeholders to organize and generate argument claims based on various social and technical factors in preparation for systematic negotiations, as well as help to reconcile design conflicts by recommending potential conflict management strategies. The conflict management strategies guide software engineers to evaluate possible alternatives from these negotiation arguments and compare these alternatives in order to choose desirable ones in the applicable circumstances. This paper also presents the design and implementation of a prototype computer supported negotiation management software system which is based on the proposed negotiation approach to support real-life software design collaboration.

Section 2 reviews the literatures relevant to this research work and discusses several underlying theoretical backgrounds for the proposed approach. The overall socio-technical argument-based negotiation management framework is presented in Section 3. It describes the integration between a socio-technical framework and argument-based negotiation process to build a socio-technical negotiation management approach for collaborative software design. The chapter explains the software design task modeling, conflict identification, design concept structure and stakeholder perspective analysis, argument building and evaluation, and conflict resolution issues. Section 4 presents the architecture and functionalities of software prototype,

called IWANT, which implements and demonstrates the application of the proposed approach. As well, some ongoing case studies and initial empiric results are described. Lastly, Section 5 summarizes the lessons learned from this research and outlines our planned future work.

2. LITERATURE REVIEWS

2.1 Collaboration and Negotiation Management in Software Design

Most of the existing negotiation research for collaborative design in software engineering falls in two categories: collaborative evaluation approaches and design negotiation management. An example of the first category is ATAM, which distributes the architectural documents and business requirements to the stakeholders, elaborates and prioritises scenarios, conducts design evaluation and finally develops a complete technical report (Kazman *et al*., 1998, Lee, 2005). It helps the engineers understand the consequence of software design with respect to the system's quality requirements and business goals. Also, it helps the developers to determine where the risks and tradeoffs exist in various software design strategies. However, as an evaluation method for software design, ATAM does not explicitly take any social factors (e.g., social interactions in teamwork) into its evaluation rationale, such as, individual goals and personal interests from the stakeholders themselves. As a result, it is not always certain that the stakeholders will accept the solutions provided by ATAM. Another software design evaluation method, developed from ATAM by the same group of researchers, is CBAM (Cost Benefit Analysis Method) (Moore 2003, Kazman 2005). CBAM collates high-priority scenarios from ATAM, and refines and prioritises scenarios to formulate business goals. It then develops architectural strategies for scenarios, calculates the total economic benefit for each strategy, and chooses architectural strategies based on business values. CBAM explores, analyzes, and makes technical decisions regarding software architecture design alternatives with consideration of economic factors (Hoh *et al*., 2002). However, it is not clear how the explored alternatives are generated from the engineers' goals and criteria, and how they satisfy the initial requirements of stakeholders with different roles, responsibilities, and priorities. As well, this approach only evaluates different design decisions but does not provide any negotiation strategy to reconcile conflicts in the software design decision-making process.

A good example of the second category is the Win-Win negotiation management approach developed by the Centre of Software Engineering at USC (Boehm *et al*., 1999). It provides a generic framework for software requirement negotiation. In the Win-Win approach, stakeholders begin by first eliciting their own desired "win conditions", identifying issues (e.g. conflicts), generating options to resolve these issues, negotiating options and finally reaching agreements. However, from the collaborative engineering point of view, in the Win-Win approach stakeholders still need to generate and negotiate the architecture alternatives manually by a rather ad-hoc process (Hoh *et al*., 2002). Furthermore, the Win-Win negotiation approach is based on a software engineering approach called MBASE (Model-Based Architecting and Software Engineering), which detects the conflict in software development by identifying model clashes (e.g. success models) (Boehm *et al*., 2002). Most of the identified clashes in this approach, however, come from past success models representing previous win conditions, which are only a subset of the stakeholders' backgrounds and expertise. Some of the potential conflicts caused by different backgrounds (e.g., technical specialties) are not accounted for because these differences are not explicitly modeled in this approach. Also, since these differences in backgrounds and expertise are often the fundamental source of the conflicts, this approach cannot easily trace where the conflict comes from even when the conflict is detected. As a result, after the present conflict is resolved, there is still a possibility that the team may be confronted with the same conflict again in future.

In the second category the most relevant work to our study is the MPARN approach, which guides stakeholders from design options to agreements by using multi-criteria preference analysis techniques (Hoh *et al*., 2002). The MPARN process begins to identify the conflicts in the stakeholders' needs following the Win-Win process and then explores resolution options. After this it supplements the Win-Win process by eliciting stakeholder preferences. It also assesses how well each of the generated options performs on stakeholder criteria. As a negotiation approach with the goal of supplementing the Win-Win process, MPARN provides some conflict analysis and resolution strategies. However, since it is based on the Win-Win process, it inherits some of the same limitations of the latter. For example, social factors (e.g. personal interests, social interactions) are not directly managed in this approach, although these factors are indispensable for real world negotiation tasks (due to the dynamical and social nature of the negotiation). The other shortcoming is that MPARN helps stakeholders analyze and prioritize their design decisions, but does not specify a negotiation management approach for the stakeholders to jointly achieve a common agreement.

2.2 Collaborative Engineering Design Approach for Conflict Resolution and Negotiation Management

Collaborative engineering design is a collection of the co-operated efforts undertaken by a team of designers and other specialists. It is involved with complicated interaction among multidisciplinary design teams in a distributed, heterogeneous and dynamic environment, including communication, cooperation, coordination and negotiation (Shen *et al*., 2001). Each team member works

on different parts of the design, from different perspectives and towards fulfilling different functional criteria.

Conflict resolution is a critical element of collaborative design because each stakeholder has her own point of view, concerns and objectives regarding the design process and hence conflicts may arise from differences in this information during multi-stakeholder interaction. In collaborative design, conflicts can be defined as "an expressed struggle between at least two interdependent parties who perceive incompatible goals, scarce rewards, and interference from the other party in achieving their goals" (Hocker *et al*., 1985). In a collaborative design context, conflicts occur when at least two incompatible design commitments are made, or when a design party has a negative critique of another design party's actions (Klein 1995).

Diverse studies in collaborative engineering design develop approaches for conflict resolution based on social sciences, artificial intelligence, computer-supported methods. Klein's research works propose a conflict resolution method based on taxonomy of conflict solving strategies mapped with taxonomy of conflicts (Klein 1994, Klein 1995). A multi-approach method for computer-supported conflict resolution is introduced by Lara and Nof providing fast conflict identification, conflict parameters diagnostics and conflict resolution mechanisms (Lara *et al*., 2003). Zhuang focuses on conflict detection by providing a method of using a web based distributed design system and intelligent agents to detect conflicts (Zhuang 1999).

More close to our research, Cooper and Taleb Bendiab proposed CONCENSUS platform and Rose proposed the CO2MED prototype to manage multi-party negotiation using computer supported conflict resolution (Cooper *et al*., 1998). Barker and his colleagues propose a tool to manage negotiation in concurrent design teams (Barker *et al*., 2001). Zhao and Deng propose an MIAS prototype to model interaction behavior including communication, negotiation, coordination and cooperation (Zhao and Deng, 2001).

In engineering design domain, a variety of theoretical models have also been built to manage design conflict. For instance, QFD (Quality Function Deployment) is a structured process that establishes customer value using the "voice of the customer" and transforms that value to design, production, and supportability process characteristics (Hauser *et al*., 1988). The result of QFD analysis is a systems engineering process that ensures product quality as defined by the customers. This is essentially a methodology to solve/mitigate the conflicts among the diversified customer needs, which mainly exist in the early phases of engineering design. The Independence Axiom in Axiomatic design states that the independence of Functional Requirements must be always maintained to reduce the random search process and minimize the iterative trial-and-error process (Suh 1990). It claims that an engineering design ignore this axiom will face substantial conflicts.

All works mentioned above suggests conflict detection mechanisms, defines conflict resolution strategies and provides support to manage the negotiation between different actors involved in the conflict. Indeed, all these works reveal that negotiation is a widely accepted approach for conflict resolution in collaborative design. However, none of them analyses the perspectives of stakeholders that are involved in the negotiation process leading to problem solving. This perspectives analysis is a critical phase for conflict resolution since conflicts originate from different perspectives.

2.3 Engineering as Collaborative Negotiation Paradigm and A Socio-Technical Framework

Real-world negotiation tasks undertaken by engineering teams are always driven by many conflicting social, economical and technical (SET) factors. Traditional engineering research has mainly focused on technical factors, with some recent efforts being extended to consider the economic factors. While recognizing the importance of both technical and economical considerations, our past research has been focusing on social factors, and more specifically, on their interactions with technical factors. We view an engineering team activity such as developing software, as a technical activity with a human purpose. Therefore, when a team of engineers (i.e., multiple stakeholders) with differing life cycle concerns come together to develop new software, it can lead to a complex socio-technical campaign. To resolve this challenge, an Engineering Collaboration via Negotiation (ECN) paradigm is developed in our past work and can best support this type of socio-technical campaign in the field of collaborative engineering.

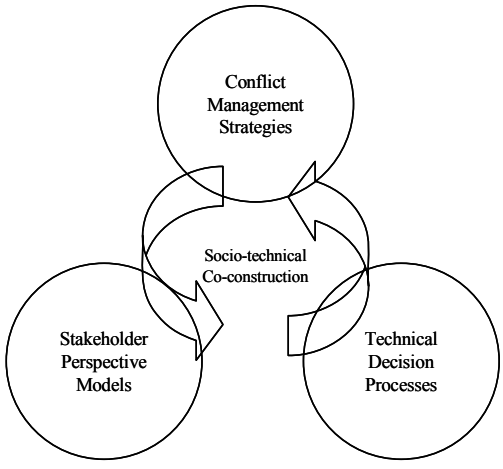

Figure 1. The key components of the socio-technical framework.

Based on this ECN paradigm, we have developed a Socio-Technical Framework (STF) as a foundation to pursue basic research in collaborative engineering. The STF has its roots in the Social Construction Theory proposed by Peter Berger and Thomas Luckmann in 1966 (Berger and

Luckman 1967), which states that meaning and institutions (e.g., a collaborative decision during software development) are a jointly negotiated and agreed construction between those participating in an endeavor. More specifically, our STF uses perspective models of stakeholders (e.g., software designers) to integrate social interactions with technical decisions, and then uses these models to manage decision conflicts during a co-construction process taken by the engineering team. As shown in Figure 1, the three core components of our STF research framework are: (1) co-construction processes taken by the engineering team, (2) perspective models of stakeholders, and (3) conflict management strategies for collaborative negotiation (Lu 2001).

The first component of our STF is the technical decision process, which serves as the baseline process for socio-technical co-construction process. The "baseline process" refers to a series of required activities, consisting of tasks and states, which must be performed by stakeholders according to some pre-established steps adapted from the specific domain practices or mandated by corporate policies.

The second component of our STF is the perspective model of team stakeholders. In our research, we define a perspective as the stakeholders' viewpoints toward specific concepts in the concept structure of a particular campaign. We also limit a perspective model to include only the purpose, context, and content of a stakeholder decision, and we only pay attention to those stakeholder perspectives that relate to specific concepts in the concept structure. In this way, we can analyze these perspective models to estimate their differences (or conceptual distances) as a measure of the degree of conflicts among stakeholders. These analysis results can provide explanations for the stakeholder decisions and/or offer rationales for conflict management by the team.

The final component of our STF is conflict management, which is the central activity of the socio-technical co-construction process. In our research, conflicts are the result of perspective model analysis and are managed systematically based on these analysis results. Conflicts are defined and quantified by the distances between stakeholder perspectives towards a specific concept in the concept structure. In our research, we detect and classify conflicts according to the composition of both concept structure and the perspective model (i.e. the sources of conflicts). And we manage conflicts by way of management of negotiation process and co-construction of stakeholders' perspectives.

3. RESEARCH FOUNDATION: SOCIO-TECHNICAL CO-CONSTRUCTION PROCESS AND TOULMIN'S ARGUMENT STRUCTURE

3.1 A Socio-Technical Co-construction Process

Based on the foundations of ECN diagram and the STF framework, we take the next step to develop a specific process through which this conceptual framework can be further detailed and made "operational" for computer implementations and engineering applications. This process, which we called Socio-Technical Co-construction (STC) process, addresses the three key components of the Socio-Technical Framework and specifies eight steps with sufficient operational details as shown in Figure 2 below.

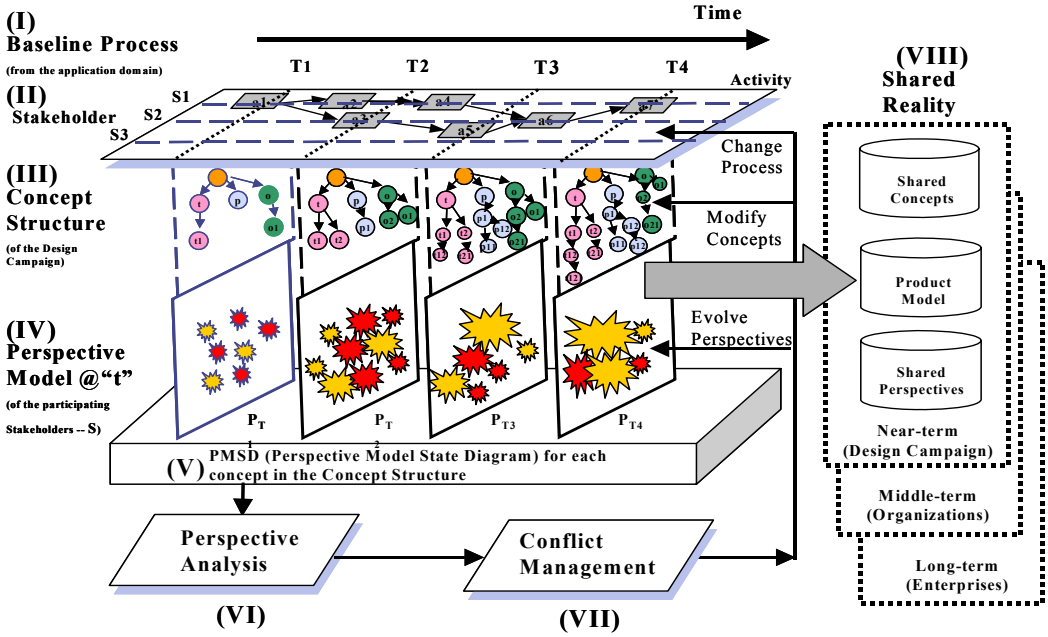

Figure 2. The socio-technical co-construction process (STC)

The Eight Steps of the Socio-Technical Co-construction (STC) Process

Ⅰ. Define a starting "**baseline process**" for the chosen application domains, as the basis to be co-constructed (i.e., changed) later, upon the agreement by all involved stakeholders, as a means to resolve conflicts. The defined baseline process can be further broken down into a series of tasks and states.

Ⅱ. Identify a group of "**stakeholders**" who have an interest in the outcomes of, and will directly or indirectly participate in, the co-construction process of a particular collaborative campaign (e.g. software design).

Ⅲ. Propose an initial "**concept structure**" for a particular engineering process to organize the concepts provided by the team.

Ⅳ. Establish the initial "**perspective model**" for all participating stakeholders to express opinions for each concept in the concept structure.

Ⅴ. Build the "**perspective model state diagram**" (PMSD) for each concept in the concept structure. A PMSD is a research apparatus in STF to depict the explicit relationships among stakeholders' concepts (including both shared and individual concepts) in addition to their purpose and context information.

Ⅵ. Perform the "**perspective analysis**" on the current PMSD. Since PMSDs link the CS with perspectives and have references to the baseline process, they provide integrated information to conveniently analyze the closeness of, or distance among, different stakeholders' perspective at that particular moment.

Ⅶ. Conduct the "**conflict management**" tasks according to the results perspective analysis.

Ⅷ. Obtain a "**shared reality**" as a result of the co-construction process. This final product of the STC process is a shared reality, which is a broader concept than traditional approaches (e.g., a finished design in terms of a product model).

3.2 Investigation of Toulmin's Argument Structure

Practicing collaborative design and negotiating dialogue have been found to be positively linked with argumentation and critical thinking skills (Hart 1990, Jennings 1998, Jin, Geslin and Lu 2005, Marttunen 1992, Smith 1977). Furthermore, the work of Buckingham and his colleagues argue that exposing an argument's structure facilitates its subsequent communication since important relationships can be more easily perceived and analyzed by others (Buckingham Shum *et al*., 1997). Stephen E. Toulmin's 1958 work, Uses of Argument, has become commonplace in structuring argumentation-Toulmin acknowledges as much in his the preface to his 1979 text, An Introduction to Reasoning (Toulmin 1979). For example, Houp, Pearsall and Teheaux's textbook, Reporting Technical Information, introduces Toulmin logic as providing "a way of checking your own arguments for those overlooked flaws. It can also help you arrange your argument" (Houp *et al*., 1998).

Argumentation is a process of making assertions (claims) and providing support and justification for these claims using data, facts, and evidence (Toulmin 1958). The goal of argumentation in negotiation is to persuade or convince others that one's reasoning is more valid or appropriate. Toulmin's model of argument provides the language symbols and data structure, which supports the argumentation process. Toulmin's model is procedural and the layout of this model focuses on the movement of accepted data to the claim through a warrant.

Toulmin also recognizes three secondary elements that may be present in an argument: backing, qualifier, and rebuttal. Backing is the authority for a warrant and provides credibility for the warrant and may be introduced when the audience is unwilling to accept the warrant. A qualifier indicates the degree of force or certainty that a claim possesses. Finally rebuttal represents certain condition or exception under which the claim will fail and hence anticipates objections that might be advanced against the argument to refute the claim (Toulmin 1958).

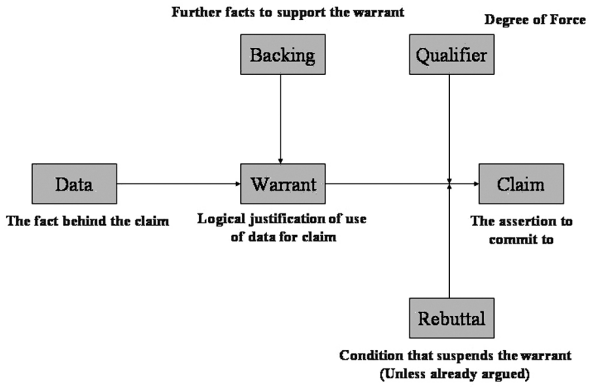

Figure 3. Toulmin's argument structure.

As such, Toulmin's argument structure becomes a mechanism for structuring argumentation between negotiating stakeholders. It aims to clarify reasoning by encouraging parties to make explicit important assumptions, distinctions, and relationships as they construct and rationalize ideas (Buckingham Shum *et al*., 1997).

We selected Toulmin argument structures to investigate negotiation after considering a number of possible approaches and structures applied or developed for negotiation (Janssen and Sage 1996). Negotiation is a process that involves both qualitative and quantitative concepts. Many of the formal approaches such as game theory and decision analysis have focused traditionally on the quantitative aspects of negotiation including goals, alternatives, and consequences. The qualitative aspects of negotiation such as argumentation, social interactions, and negotiator perspectives are typically given low priority and are sub-

sumed in other quantitative concepts, although they are important in their own respect from a practical viewpoint because in many cases negotiation approaches directly impact the negotiation outcome. The negotiation approach proposed in this work will capture these qualitative aspects of negotiation through the use of argumentation methodologies.

Other types of structures than Toulmin's argument include: influence diagrams and Bayesian network, which are oriented toward quantitative analysis and are not easy to build; structured modeling and conceptual modeling, which may be used to construct a Toulmin argument structure; and Wigmore structured evidence, which comes the closest to Toulmin and is an argument structure based on an elaborate set of symbols (Janssen and Sage 1996). The Wigmore approach, although closest to our Toulmin based approach, is much more complicated than The latter. The Toulmin argument structure is easy to use and understand and is widely applicable (Janssen and Sage 1996). Our studies suggest that, when the argument structure is framed appropriately, it is easy for software designer to understand and to use in negotiating design solutions. Many of these approaches are described in the contemporary literature (Sage 1992, Schum 1994). There have also been a number of other studies concerning the implementation of Toulmin based logic (Lagomasino and Sage 1985, Sage 1991, Laskey 1989).

Also Jassen and Sage's study shows there are several advantages to a graphical or structured depiction (i.e. boxes and lines as in the Toulmin's argument structure) than natural language description (Janssen and Sage 1996). They have stated the reasons like follows: first, visualization eases comprehension. The components of the argument are explicitly represented, meaning that it is easier to identify the particular elements of an argument, and these elements of the structure provide stakeholders for the elements, thereby facilitating elicitation of these elements. Second, it is easy for the person filing the boxes to see what is missing as well as the reasoning that has been put forth. In this regard it is easier to compare arguments between multiple experts, and between claim and counterclaims than between statements in generally unstructured discourse (Janssen and Sage 1996).

It is true that using the Toulmin's argument structure, which is generally more objective than implicit arguments, it is hard for stakeholders to hide bias because the grounds and backing of an argument are clearly listed and described to support the claims. Therefore, all stakeholders' perspectives are generally relatively easy to be observed by others through examination of the ground and warrants that the stakeholder expresses (Janssen and Sage 1996). As mentioned earlier in this section, argument structure has been used to build an argument-based negotiation process model in many studies including collaborative design previously and proved to facilitate more objective and fair communication (Chang *et al*., 1995, Sillince *et al*., 1999, Jennings 1998, Amgoud *et al*., 2000, Avery *et al*., 2001, Kraus, 2001, Rong *et al*., 2002). However, there are remaining unresolved issues in most of the above work, such as a systematic guide to compose the arguments based on the information from particular domain or process and an evaluation method on the arguments to obtain a most desired one.

In this work we have proposed the use of Toulmin's argument structure to support the negotiation management in a socio-technical argument-based negotiation process.

4. AN ARGUMENT-BASED SOCIO-TECHNICAL NEGOTIATION (ASTN) APPROACH

4.1 Definitions of Terminologies

This work focuses on managing negotiation tasks in which a team of software engineers must collaborate with each other to design a software solution. The subject of collaborative negotiation in software engineering is part of an emerging research field, called collaborative engineering. In this new research field, collaborative engineering is defined as a socio-technical group decision-making process, whereby a team of engineers collaborate to resolve conflicts, bargain for individual or collective advantages, agree upon courses of action, and/or craft joint decisions that serve their mutual interests. Unlike traditional engineering tasks, which are often treated as a purely technical decision-making process of "task-work" by an individual, collaborative engineering tasks are, additionally, a social endeavor of "teamwork" by a team of individuals. In practice, collaborative engineering is best carried out in a "team" environment where, unlike a "work group", all team members have already agreed on a common goal to achieve. Based on this belief, software design is essentially a human-based, interdisciplinary teamwork, and must be modeled as socio-technical collaboration accordingly.

In our research, "social" refers to the behaviors that take the interests of others and the common stakeholder characteristics, which influence collaborative team dynamics during social interactions such as background, objective, and preference. The dominant characteristics of software designers (engineer, architect, managers, etc.) tend of be such issues as choice of programming languages, preferred development methodologies, expectation from project work and design team, and career goal in the organization. They are initially brought into the collaborative teamwork by the participating stakeholders, and then continuously co-constructed and evolved during the social interaction process. Based on the above meanings, the term "socio-technical" signifies the mutual consideration of and the true integration between the social (teamwork) and technical (task-work) aspects of engineering activities.

In order to manage the social interaction in the design team, in our research we also define the "perspective" as the particular attitudes (i.e., viewpoints) via

which the stakeholder views their own concepts and others' when making decisions, e.g. strong desire on own objective, support or disagreement on other's objective. Furthermore, the conflict needs to be defined in this work: its general definition is "an argument about something important or a state of opposition between persons in idea or interests" and specifically in our research, conflict is defined as the argument between stakeholder about the design tasks and this argument roots at difference in stakeholders' perspectives towards their understanding and expectation of the design task. The conflict is therefore identified in the design task and to resolve it in collaborative design requires the explicit modeling of stakeholder perspectives.

In summary, the above definitions in our research explicitly acknowledge collaborative design tasks as a dynamic interface between individual decisions and group interactions, and as an assimilation of social and technical activities operating in parallel over different time, space, and discipline scales in a software engineering team.

4.2 An Argument-based Socio-Technical Negotiation Approach

In the collaborative design of software engineering, a team of stakeholders with different social backgrounds and technical expertise must jointly undertake many common tasks, which require making joint decisions based on mutual agreements. During this process, stakeholders often have a variety of opinions and therefore must negotiate with each other to arrive at a shared understanding about critical issues at hands. They must make many common decisions to develop design solutions for the software in despite of any conflicts caused by the social and technical differences. Therefore, a specific challenge in this process is to help the team members reconcile these differences, resolve the conflicts in their decisions, and achieve common understanding about the design solutions.

To address the above challenge, this work proposes an Argument-based Socio-Technical Negotiation (ASTN) approach to manage the negotiation activities in collaborative design process in software engineering. Traditionally, software engineering approaches often ignore social interactions and treat collaborative design as a purely technical problem. As a result, decision analysis and negotiation approach are solely based on technical considerations-all social interactions are implicitly dealt with in an ad-hoc manner. The inability to model the human perspective and social interaction as an integral part of technical decisions is a major roadblock to resolving conflicts in collaborative design. We believe a collaborative design process is not only the technical decision making process but also a social interaction process amongst the members of the design team. Based on this belief, ASTN approach overcomes the limitations of traditional work by explicitly modeling both the social and technical factors in argument structures and then systematically guiding the team through a negotiation process in which their differences in technical decisions are reconciled. The following sections give an overview of this approach and Section 4.3 discusses it in greater details.

4.2.1 Overview of the Socio-Technical Negotiation Process

Socio-technical negotiation process is the main strength of this approach. It describes how the stakeholders pre-

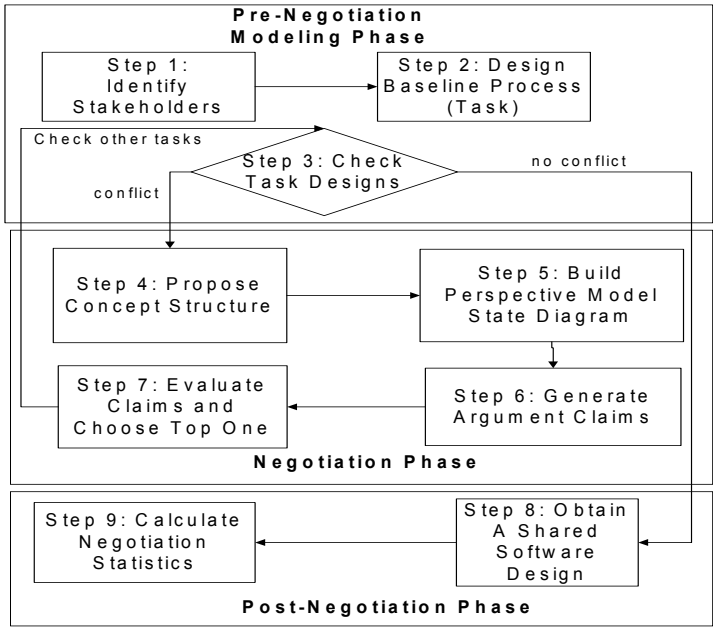

Figure 4. A socio-technical negotiation process for collaborative design in software engineering.

pare the negotiation arguments using both social characteristics and technical decisions and then carries out the negotiation process in order to reach an agreement. Figure 4 illustrates this integrated approach, which has three inter-related phases.

First, the Pre-negotiation phase starts with the baseline process of software design. This baseline process captures the required technical task-works in a predetermined order. For example, it can be a commonly accepted "workflow" suggested by the domain experts or standard operating procedures instituted by the company. This approach also needs to identify a set of "stakeholders" who have an interest in the outcomes of, and will directly or indirectly participate in, the co-construction process. It starts the design process, asks each stakeholder to propose an implementation for specific tasks in the design process, and checks the differences (i.e., conflicts) between proposed implementations. Then it determines whether to initiate a negotiation by identifying conflicting implementations of a design.

Second, the Negotiation phase helps the stakeholders prepare for their arguments, and guides them into resolving the conflicts by an argument-based process. In this phase the stakeholders jointly propose a concept structure to represent all relevant factors (e.g. background, objective, etc), which influence the design task and declare their perspectives upon the concepts. Then based on the design tasks, concepts and perspectives, the stakeholders are systematically guided to build their negotiation arguments. The stakeholders review the arguments of each other and may be suggested to change concept, perspectives or even implementation of the design task. They will reach an agreement during the negotiation or their argument will be compared using an argument evaluation method in order to choose a most desirable one for the team.

Lastly, the Post-negotiation phase uses two inter-related steps to assure that the stakeholders obtain a commonly accepted software design implementation, and tracks relevant collaboration performance statistics (e.g., negotiation time used by the stakeholders). These statistics are useful for future references by the design team in case of similar tasks and/or conflicts. Section 4.3 explains these three ASTN phases in more details.

4.2.2 Integration of Socio-Technical Negotiation Process and Argument Structure

The "argument" in the ASTN approach is built based on the Toulmin's structure of argument (Toulmin 1958). As discussed in Section 2.2, the Toulmin structure is mostly procedural, and its layout focuses on the movement of accepted data to the claim through a warrant

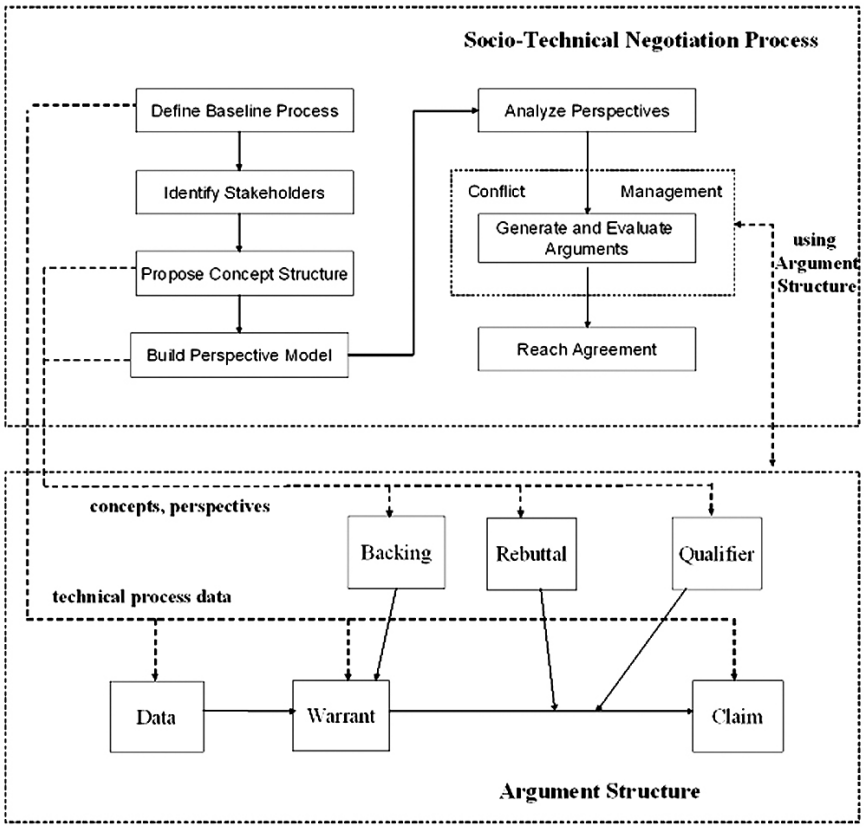

Figure 5. Integration of STCP and ABNP

(guarantee). The claim states the current "position" that the stakeholder commits to about the issue being argued. The data is the "evidence" behind the claim and the "warrant" logically justifies the use of the data for that specific claim. Also there are three secondary elements that may be present in an argument: backing, qualifier, and rebuttal. Backing is the authority for a warrant, provides credibility for the warrant, and may be introduced when the audience is unwilling to accept the warrant. A qualifier indicates the degree of force or certainty that a claim possesses. Finally, rebuttal represents certain conditions or exceptions under which the claim will fail and hence anticipates objections that might be advanced against the argument to refute the claim.

Note that argument structure, by itself, stays as a general guidance for negotiation and does not specify how to obtain the information for each component in its arguments. ASTN then uses this structure to organize the critical information for negotiation based on both social and technical factors. On the technical side, the baseline design tasks model the stakeholders' decisions for the software design and concept structure helps stakeholders share their understanding of the design tasks in terms of concepts. On the social side, the concept structure helps the stakeholder declare their characteristics (e.g. background, values, etc), which have a great impact on the technical decisions. Based on these characteristics, the perspective model can represent the social interaction of the stakeholders. Figure 5 describes a mapping scenario between the data flow in ASTN and the argument structure. As shown in the figure, the technical factors in the ASTN provide the argument structure with the major elements (e.g. claim, warrant, and data) in the argument data structure. The social factors correspond to the secondary argument elements (e.g. backing, qualifier, and rebuttal). Based on these factors, in the step to manage the conflicts, ASTN uses the argument structure to help stakeholder prepares for the systematic negotiation for conflict resolution.

4.2.3 Arguments Evaluation

Another key feature of this approach is the argument evaluation method. At the end of negotiation, if all the stakeholders can jointly agree on a particular implementation, they can take that as the conflict resolution. Otherwise, all the arguments must be carefully evaluated for resolutions. This evaluation method analyses the perspectives and concepts from stakeholders and compares the argument claims based on the analysis result. To build this method, we have investigated the field of multiple objective decision analyses (also known as Multi-Criteria Decision Making, MCDM). The focus of MCDM is usually on a single decision-maker who unilaterally chooses between alternatives whose outcomes differ on two or more "objectives" or "criteria" (Yoon and Hwang 1995). In ASTN, the "objective" and "criteria" are the values of each stakeholder, which are stored in concept structure and included in the arguments.

In MCDM there have been a variety of strategies developed to calculate the best choice for a decision making problem. Optimization selects the first alternative with best value for single criterion or aggregate measure. It also establishes a set of constraints, which are the requirements on the alternatives. Depending on the decision-making problem, different optimization techniques can be applied, linear programming, non-linear programming, discrete optimization, etc. (Nemhauser *et al*., 1989). Multiple criteria optimization applies for the case having a finite number of criteria and an infinite number of the feasible alternatives (meeting the requirements) or finite number of feasible alternatives but only in implicit form (Steuer, R. E. 1986). Satisficing strategy eliminates all alternatives that fail on one constraint (involving one or more criteria) and then consider another constraint. Mixed scanning strategy decomposes the overall decision objective into a hierarchic structure of criteria and alternatives and build pair-wise comparison matrix to calculate relative weights of the criteria which are then used to arrive at a score for each alternative (Etzioni 1967). Another strategy, Analytical Hierarchy Process (AHP) a pairwise comparison method developed by Saaty (Saaty 1980) to assign the quantitative weights to criteria and scores to alternatives. Statistical technique, also known as deterministic value function, ranks the alternatives from best to worst and assume one can characterize the consequences of each alternative with certainty or that only their expected value matters. Utility function methods are similar to deterministic value function methods, except that they consider uncertainty in the criteria values for each alternative (Anderson *et al*., 2002).

Currently ASTN uses the Deterministic Value Function from MCDM field to evaluate the values and choose a most desirable argument as solution. It is chosen because of the nature of software design in that there are finite alternatives (implementation) for one conflicting design task (decision-making problem) and we can fairly assume the designers (decision makers) with different background and expertise, who implement this task, can understand the consequences of each alternative with certainty and can also well describe their expected value which are their fundamental objectives. Deterministic value function methods provide a ranking of the arguments (from best to worst) as group preference based on the stakeholders' preferences. It is important to note that in the domain of Social Choice Theory, Arrow suggests that it was reasonable that the relationship between individual preferences and social preferences should have the following properties: Pareto optimality, non-dictatorship, unrestricted domain, rationality and independence of irrelevant alternatives. Arrows showed that there were no social choice functions, which satisfied all five properties (Arrow 1951). However, Arrow's results make rather weak assumption about individual preferences – he assumes cardinal non-comparability of individual preferences. By making stronger assumption of cardinal comparability, i.e. assuming stakeholder can characterize their

expected value of each argument and can present certain degree of desire over each value, which is generally the case of software design undertaken by a group of design experts, we can broaden the class of choice evaluation functions.

Another concept, which we have investigated and helps us justify the ASTN approach, is "bounded rationality." It was developed in modern organizational behavior research (Simon 1957). The concept does not reject the goal of achieving collective rationality, but it acknowledges the always existing limits on the degree of rationality which can be practically accomplished despite of the strong desire of people to be rational. These limits include limitations of the human mind, the structure within which the mind operates and cognitive inability to process and compute all alternatives. Because of these practical limits on rationality, instead of optimizing decision to be absolutely rational, the best which decision makers can do is to sacrifice, i.e. to choose the first option that meets given need or addresses most needs, instead of seeking for an absolutely optimal one. This acknowledgement is important to our work since in reality, due to changing needs and limited resource, engineers often try to choose the outcome having maximum chance of being satisfactory instead of pursuing absolutely best choice. Although the outcome of the argument evaluation method in this work is a tentative "best" to "worst" ranking of arguments, yet the appropriate use of the method is to gain group insight on all arguments and choose a most desired one by the group judgment rather than mechanically to pick a supposedly "best" argument. The equation below describes the deterministic value function in general and the evaluation function in ASTN will be explained in Section 4.3.

$$\text{MAX } V(Xj) = \sum_{i=1}^{I} w_i v_i(x_{ij}) \quad (1)$$

Where:
V(Xj) = overall value of alternative *j*,
Wi = weight for criterion *i*, i.e. perspective for concept i, given by stakeholder
I = number of criteria

This technique uses single attribute value functions Vi(Xij) and weights to obtain an overall "value" for each argument. Inside the equation, weight is the perspective declared by the stakeholder for a particular concept.

4.3 Phases of the Argument-based Socio-Technical Negotiation Process

4.3.1 Pre-Negotiation: Design Task Modeling and Conflict Identification

The goal of the pre-negotiation phase is to identify all potential conflicts by checking the differences between proposed task implementations, and helps the stakeholders manage their argument information in order to negotiate the identified conflicts. For example, in a common software design task "estimate quality attributes", team members often have different views. Salespersons may suggest performance and usability as the most important attributes; while engineers may argue that maintainability is most important for the long run. Meanwhile, project managers may believe that portability is critical, due to possible future options to migrate the software to a variety of the operating platforms. We will use this example to explain how a design conflict will be identified and relevant information will be modeled in this section. There are five specific steps in this pre-negotiation phase of our ASTN approach.

Step 1: Identify the "*stakeholders*" who participate in the software design team via ASTN.

Stakeholders are those software design team members who have an interest in the process and/or outcomes of the software design decisions (i.e., implementations) and may directly or indirectly participate in the STCP.

Step 2: Prescribe a "*baseline software design process*" to initiate the STCP.

A baseline software design process is defined as a series of necessary technical task-works that must be undertaken by the team to develop a software design solution. ASTN takes this design process as the baseline to begin the STCP process. This process and its associated standard design task-works are generally pre-defined based on the domain practices or chosen for the stakeholders by the management, e.g. Object-Oriented Design Process, which comes with a set of standard procedures. Also in this step the software corporation must set up common goals for software design team to achieve and define a set of shared values that all team members follow during the design activities. Goals and values set the direction for the design team to identify and define their decision objectives in the negotiation phase.

Step 3: Ask stakeholders to implement the above design tasks and check the difference in their implementation details.

Although stakeholders jointly work on the design tasks according to the baseline process prescribed above, due to their divergent background, interest, experience, and expertise, they will undoubtedly come up with different technical decisions in the implementation details of these tasks. In the ASTN approach, the implementation of a software design task is defined as a logical sequence of actions/objects, combined with necessary resources including time and staff. For example, regarding the example software design task "estimate quality attributes", a possible implementation proposal can be specified as:

{objects: performance, security and usability; actions: estimate performance, security and usability according to the functional requirements; resources: the

design team work for one day}.

Therefore, if there are different decisions, objects or resources in the implementations proposed by the stakeholders for a specific design task, the team will declare a conflict. A typical conflict could be, for example, that the stakeholders are using different objects. Just like the example mentioned at the beginning of this section, the objects (i.e. quality attributes) for the task are different amongst all the stakeholders. In case of a conflict, the process will continue to the Negotiation phase next to develop a mutual agreement for resolving the conflict. Otherwise (i.e., no conflict), the process will move forward directly to the Post-Negotiation phase (see Section 4.3.3) with supporting agreements on how to implement all design tasks.

4.3.2 Negotiation: Argument Generation and Evaluation

In this phase, the participating stakeholders are guided to negotiate with each other by an argument-based process based on ABNP until a mutual agreement is reached. In most instances a general argument-based negotiation process is managed by undertaking the following two stages:

- Stakeholders generate argument claims (or counter proposals) for concerned issues and provide supporting data, and
- Stakeholders exchange and respond to others' claims (or counter claims) and their associated supporting data (Sierra *et al*., 1998)

The above two-stage argument-based process is used to resolve the conflicts during this phase. What is new in our ASTN approach is the building of more comprehensive argument structures by including both the social and technical factors. The technical factors, such as the baseline process and design task implementations, are obtained as part of the STCP. The social information, such as the objective to undertake the task and the criteria to design the solution, is extracted in the first two steps (Steps 4 and 5) below. The whole negotiation phase of ASTN is composed of four steps (4, 5, 6 and 7) as follows:

Step 4: Propose a "*concept structure*" for the identified conflicting design task.

Having conflicting implementation of a design task in the baseline process indicates some differences in the social factors about the stakeholders. These differences may root from the social (i.e., non-technical) characteristics of the stakeholders, which impact their technical decisions and evolve during the social interaction and team collaboration. These characteristics include, as discussed earlier, the fundamental objectives for which the task is undertaken, the mean objective which helps achieve the fundamental, and the criteria how the fundamental objectives are achieved by the implementation. Based on the goals defined at the beginning of the design process, the software team should be able to identify the values and objectives in this step.

To better capture these differences and get a deeper understanding of the conflict and these characteristics, the ASTN approach provides a structure, which models the concepts that underline the proposed technical decisions and represent their understanding and expectations ("values") of the design tasks. This concept structure is a model to organize these social factors perceived by the individual stakeholder. This model is proposed by the stakeholders based on their separate perceptions of the conflicting design task, and the concepts in this model will be dynamically changed by the social interactions among the stakeholders. In reference to the information in a concept structure, the stakeholders can declare their opinions (e.g., how much they support others' concepts) regarding this design task and the differences causing the conflict can be identified.

To explain the concept structure further, we continue to use the software design task "estimate quality attributes" as an example. Table 1 below describes this example concept structure, including information about stakeholders, objectives, and criteria. There are three stake-

Table 1. An example concept structure for "estimate quality attributes."

Stakeholder	Fundamental Objectives	Criteria	Mean Objectives
Salesperson	Performance is first priority, especially response time. Security and Usability should also be guaranteed.	Sale is most important. All quality attributes should be measured by sale requirements first.	Product needs to beat competitors in performance in order to achieve good sales. Other considerations should yield to this.
Engineer	Performance, easy-to-maintain, security, and usability.	The quality attributes should be determined based on appropriate development resource and proof-of-concept.	Build a prototype to get software quality statistics.
Manager	Portability, performance, security, and usability.	Project responsibility and executive decisions.	Deploy the development environment on different platforms to ensure portability.

holders in this example: salesperson, engineer, and manager. Salesperson's fundamental objectives are to guarantee performance, security and usability of the software and mean objective is to at least ensure the product is better in quality than the competitors on the market. Her criteria are that sale is most important and hence every attribute should be evaluated by sale requirements and marketing strength. The engineer, on the other hand, believes that software performance, maintainability, security, and usability are most important, which could be achieved by building a prototype system to test those attributes, and all decisions must be based on these criteria. Meanwhile, the objectives of the third stakeholder, manager, include performance, security, usability and portability and his criteria may also include project responsibility and other executive decisions.

Step 5: Establish a "*perspective model*" for each stakeholder based on the above concept structure.

Once a concept structure is established by the team, all stakeholders can express their own opinions (i.e., claims about an implementation) via the social interaction process in STCP. Social interaction is a very complex human phenomenon in teamwork that consists of many interrelated psychological and organizational factors. There is no practical way that a complete modeling of social interactions can be fully developed and incorporated. As a result, our ASTN approach takes a rather simplified view toward social interactions by focusing on modeling the dynamic impacts of social interactions on the evolving "perspectives" of the stakeholders as they express their opinions toward the concept structure. These dynamically evolving perspectives are represented as a "perspective model" for the said concepts of which the stakeholders have the common interests or some expertise. In other words, the perspective model dynamically depicts a stakeholder's perceptions of his or others' concepts. These perceptions could include the stakeholders' desire for their ideas to succeed and their support for or disagreement with the concepts of others. Therefore, the perspective models indicate the difference in stakeholders' opinions, which cause the conflict in the technical implementation of the tasks. And these models will be further analyzed next to systematically reconcile the conflicts in our

Table 2. Perspective model-stakeholders' desire for own fundamental objectives.

Stakeholder	Objectives	Desire (1 to 5)
Salesperson	Performance Security Usability	Performance: 5 Security: 4 Usability: 4
Engineer	performance, easy-to-maintain, security, and usability,	Performance: 5 Easy-to-maintain: 4 Usability: 4 Security: 4
Manager	performance, security, usability, and portability	Performance: 5 Security: 5 Usability: 4 Portability: 5

Table 3. Perspective model - stakeholders' support or disagreement for others' fundamental objectives

Stakeholder	Sales	Engineers	Manager
Salesperson	n/a	Performance: 5 Easy-to-maintain: 3 Security: 5 Usability: 5	Performance: 5 Security: 5 Usability: 5 Portability: 3
Engineer	Performance: 5 Security: 5 Usability: 4	n/a	Performance: 5 Security: 5 Usability: 5 Portability: 3
Manager	Performance :5 Security: 5 Usability: 4	Performance: 5 Easy-to-maintain: 2 Security: 5 Usability: 5	n/a

Table 4. Perspective model-stakeholders' desire for own criteria.

Stakeholder	Criteria	Desire (1 to 5)
Salesperson	Sale requirement	Sale requirement: 5
Engineer	Development resource	Development resource: 4
Manager	Project responsibility and executive decisions	Project responsibility: 5 Executive decisions: 4

Table 5. Perspective model-stakeholders' support or disagreement for others' criteria.

Stakeholder	Sales	Engineers	Manager
Salesperson	n/a	Development resource: 3	Project responsibility: 4 Executive decisions: 5
Engineer	Sale requirement: 3	n/a	Project responsibility: 4 Executive decisions: 5
Manager	Sale requirement: 3	Development resource: 3	n/a

Table 6. Perspective model-stakeholders' desire for own mean objective.

Stakeholder	Mean Objective	Desire (1 to 5)
Salesperson	n/a	n/a
Engineer	Software prototype	Software prototype: 4
Manager	Platform Portability	Platform Portability : 4

Table 7. Perspective model-stakeholders' support or disagreement for others' mean objective.

Stakeholder	Sales	Engineers	Manager
Salesperson	n/a	Software prototype: 4	Platform Portability: 4
Engineer	n/a	n/a	Platform Portability: 4
Manager	n/a	Software prototype: 3	n/a

negotiation approach.

Although stakeholder perspectives are often highly subjective in nature, some quantitative methods are needed in order to further analyzed the perspective models. In our research, we use a 1-5 scale to quantify the stakeholder's desire for his own concept and support (or disagreement) for others' concepts. In order words, when expressing the perspectives (i.e., opinions) for each concept in the concept structure, the stakeholders use a 1-to-5 scale where the ranking is as follows:

For personal concepts:
1 = undecided
2 = least desire
3 = slight desire
4 = desire
5 = strong desire

For other stakeholders' concepts:
1 = strongly disagree
2 = disagree
3 = undecided
4 = agree
5 = strongly agree

For example, in the above "estimate quality attributes" design task, a salesperson strongly desires performance most, security second, and usability third. So, for her own concepts, her perspectives are represented as {performance: 5; security: 4; usability: 4}; the salesperson strongly agrees with the engineer on performance, security, and usability, but not on ease-of maintenance. So, for the engineer's concepts, her perspectives are represented as {performance: 5; security: 5; usability: 5; easy-to-maintain: 3}. Accordingly, Table 2 to Table 7 show some example perspectives of the stakeholders for the example design task "Estimate Quality Attributes". Table 2 and Table 3 show the perspective models for their personal concepts. Table 4 to Table 7 show the perspective models for the concepts of others. Positive numbers indicate support, negative indicate disagreement.

Step 6: Facilitate the generation of stakeholder negotiation arguments, including claims and the supporting data.

In order to model both social and technical factors in stakeholders' negotiation arguments, the claims, data and warrants are collected from the baseline process repre-

senting the technical decisions. And backing, qualifiers and rebuttal are obtained from the concept structure and stakeholders' perspective models, which jointly represent stakeholder's social characteristics and their social interactions. Based on the definition of Toulmin's structure, the claim is the proposal of the argument. In our ASTN approach it is how a stakeholder proposes to implement the design task in terms of the sequence of the actions/objects. The data consists of the initial state and expected state of the task. The warrant is the logical relationship between the task and the states. Therefore, the data actually validates the claim and the warrant justifies the use of the data for the claim. Backing and rebuttal comes from the concept structure. The objectives and criteria of the stakeholders are the backing information, which logically supports the warrant. And in ASTN approach, the rebuttal in the argument is actually filled in by other stakeholder (than the original owner of the argument) in case of a counter-claim and it provides other options, which suspends the warrant and put against the claim. Stakeholders' perspectives (e.g. desire, support or disagreement of a concept) are the qualifiers, which indicate the degree of force to validate his claim.

To build the negotiation argument in this way, stakeholders have a better understanding of each other because they share not only their claims but also their underlining reasons and desires (e.g., perspectives). Figure 6 below describes an argument example from a salesperson stakeholder's perspective. As shown in the figure, the claim for the task "estimate product quality attributes" is to test response time, data security and software usability. The data describes that the initial state (of this task) is that the functional architecture draft is ready for review and the expected result is that the quality of the architecture should be well evaluated and validated. Therefore, it is critical that, within this task, all quality attributes (with which the customer is concerned) are estimated. To justify the use of the data, the warrant states the importance of the latter in validating the claim. The backing of this argument is to present the salesperson's objective (i.e. performance, security and usability) and criteria (i.e. sales requirements are first-priority). Extended data from customer requirements is also provided in the backing. The qualifier is the salesperson's perspective, i.e. her desire for the performance, security and usability. The qualifier also includes her support or disagreement on other's concepts.

Step 7: Exchange the arguments among the stakeholders and compare them by an argument evaluation approach.

As the stakeholders share and exchange their argument claims during negotiation, their concept structures and perspective models may evolve due to social interactions and/or deepened understanding of each other. If all the stakeholders can jointly agree on a particular claim, they can take that claim as the conflict resolution. Otherwise, all the arguments must be carefully evaluated for resolutions. The evaluation method analyses the stakeholder perspectives of the concepts within the arguments and compares the argument claims based on the result. The stakeholders can choose a particular evaluation method according to their requirements. In this work, a simple example is provided, as an illustration, using "weighted average" to evaluate the arguments based on concepts and perspectives, e.g. objectives, criteria, support, disagreement. Weighted average, by its definition, means an average that takes into account the proportional relevance and strength of each component, rather than treating each component equally. In our ASTN approach,

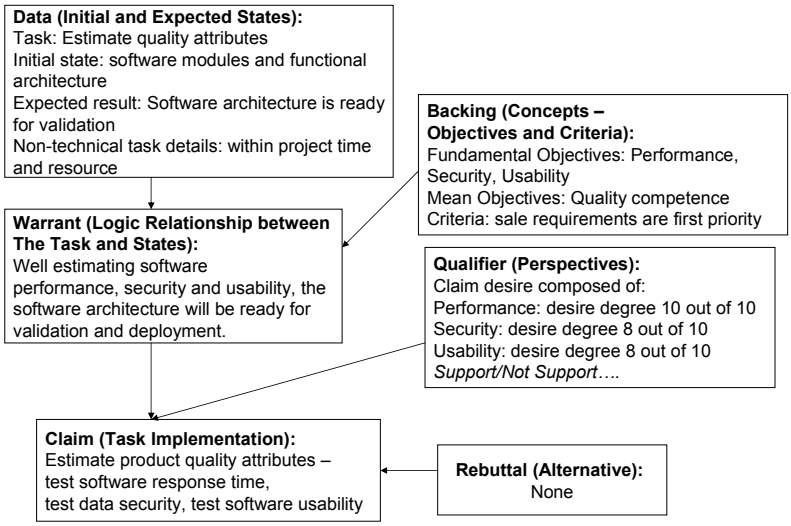

Figure 6. An example argument (by the salesperson).

the component is the concept and the relevance value is the perspective. As explained above, they are initially proposed in the pre-negotiation phase and then evolve in the negotiation phase during the argument exchanges. This specific evaluation method is explained as follows:

- The evaluation result for an argument clam is the sum of the results for fundamental objectives, criteria and mean objectives. The mathematical formula of this method is as follows:
  Claim evaluation = average (fundamental objectives evaluations + criteria evaluations + mean objectives evaluations)
- The method to evaluate a fundamental objective is to add the sum of the desires for this fundamental objective and the support or disagreement from others. The evaluation of this fundamental objective is represented by the "weight" (i.e. perspective) added by all the stakeholders. Its mathematical formula is as follows:

$$\text{Objectives evaluations} = \frac{\sum (objective\_desire + \sum_{stakeholder\ r} support / stakeholder\ r\_number + \sum_{stakeholder\ r} dissent / stakeholder\ r\_number)}{\sum objective\_number} \quad (2)$$

With this formula, we can evaluate an objective of a stakeholder by adding his desire with the support or disagreement from others. Each other stakeholder's support for this objective is summed up and divided by the number of the stakeholders. Same calculation applies for the disagreements. Given the evaluation for each objective, we can calculate the overall evaluations for the objectives of a stakeholder by adding all evaluation value together and dividing the sum by the number of the objectives. The method to evaluate the criteria is listed below and the method to evaluate mean objectives is same as that for the fundamental objectives.

$$\text{Criteria evaluations} = \frac{\sum (criteria\_desire + \sum_{stakeholder} support / stakeholder\_number + \sum_{stakeholder} disagreement / stakeholder\_number)}{\sum criteria\_number} \quad (3)$$

For example, the evaluation for the manager's claim is:
Claim evaluation
= average (fundamental objectives evaluations + criteria evaluations + mean objective evaluations)
= [(performance_evaluation + security_evaluation + usability_evaluation + portability_evaluation) + (project_responsibility_evaluation + executive_power_evaluation) + (platform_portablility_evaluation)]/3
= [[(5+5+5)/3 +(5+5+5)/3+(4+5+5)/3+(5+3+3)/3] /4 + [(5+4+4)/3+(4+5+5)/3]/2 + (4+4+4)/3]/3
= 4.33

Following this same example, Table 8 shows the evaluation results.

Table 8. Arguments Evaluation Results.

Stakeholder Claim	Evaluation
Sales	4.07
Engineers	3.73
Manager	4.33

After the evaluation, the stakeholders choose the argument claim with the highest score and move back to Step 3 above to check for further conflicts with other tasks. These iterations continue until no more conflict is found, and the team moves to the Post-Negotiation phase as described below.

4.3.3 Post-Negotiation: Shared Reality and Collaboration Statistics

In the Post-negotiation phase, the stakeholders have resolved all identified conflicts and are committed to accept one jointly agreed software design. After the stakeholders have completed all the design tasks in the baseline process and the necessary negotiation activities are well managed, and they have converged onto one common software design, the collaboration statistics are calculated and summarized. There are two steps in this last phase as described below.

Step 8: Obtain a commonly accepted software design.

One outcome of the ASTN approach is a software design commonly agreed and accepted by all involved stakeholders. No conflict exists for this software design. In addition, it also includes the shared concepts and common understood perspectives, which have been collected during the previous negotiation phase – they can be very useful for negotiation management in future collaboration among the same group of stakeholders on similar software design tasks. The concept structure built in the negotiation process can also provide a clear explanation of the software architecture and functionality for other teams (e.g. software quality assurance) to learn and coordinate in large software projects.

Step 9: Collect and report the collaboration statistics.

Collaboration statistics include, for example, negotiation efficiency (i.e. how much time or how many iterations were spent on resolving one conflict), the number of the conflicts (which are detected and resolved) and conflict profiles (what are involved social and technical factors). These collaboration statistics are calculated as the summary of the past stakeholder negotiation activities and will be an important factor for evaluating the quality of collaboration in software design process. For example,

less number of conflicts or less times of iterations normally indicates better collaboration. Therefore, the statistics will help team managers further refine the task-work (prescribed in the baseline process) by investigating the specific steps in the negotiation process that are most time-consuming or causing unexpected iterations. Additionally, the negotiation efficiency and the conflicts profiles will be available as useful future references for the design team management in case they face with similar conflicts in the future.

5. PROTOTYPE IMPLEMENTATION AND SYSTEM EVALUATION

Using the Argument-based Socio-Technical Negotiation approach (ASTN) approach presented in the previous section, in this work we are developing an Intelligent Web-based Argument Negotiation Toolkit (IWANT). IWANT is a computer-supported negotiation process management system based on the ASTN approach, the socio-technical argument-based negotiation process for the collaborative software design. The unique contributions of this system are in two folds: the modeling and analysis of social interactions and technical decisions of the stakeholders, and the management of a negotiation process based on the integration of Socio-Technical Co-construction Process (STCP) and argument structure. It provides a toolkit to help the stakeholders to systematically carry out a socio-technical negotiation process to resolve their decision conflicts in collaborative software design. This section briefly explains the functionality and architecture of the research prototype IWANT and its application-a case study of negotiation process management in software design using IWANT.

5.1 System Functionality and Architecture Design of IWANT

The major functionality of IWANT is to help stakeholders to systematically negotiate their design conflicts based on both technical and social factors and well manage this negotiation process by providing a step-by-step procedure based on the STCP. When stakeholders realize that there are different implementations in their software design tasks, they can activate the negotiation process by logging into the IWANT system and launching a new process instance. The new instance first collects argument information from the stakeholders by requesting their concept structure and perspective models. After that, the IWANT system shares the argument claims within all members in the design team. It can also track the evolving concepts and stakeholder perspective, and make relevant changes in the negotiation arguments. If the stakeholders cannot choose a claim by themselves, IWANT can provide them with a few evaluation approach options (e.g. weighted average) and then evaluate all the claims using the approach chosen by the stakeholders. After that, the team takes the claim with the best score and continues their design work. To manage the negotiation process in collaborative software design, IWANT also has the ability to model a generic software design process (i.e., the baseline process) and build stakeholders profiles, which can be provided for review during the negotiation.

Based on the specified functionalities, Figure 7 shows the overall system architecture of IWANT.

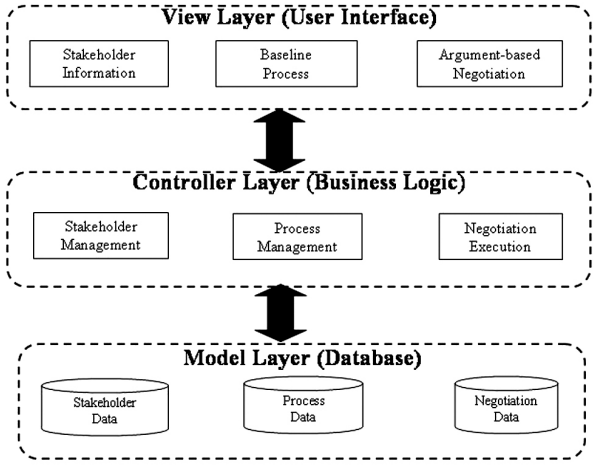

Figure 7. The IWANT system architecture

In more details, the system is designed based on three layers, in accordance with widely accepted Model-View-Controller standard (Alur, 2001) in system design domain:

- The view layer is the user interface component running on the client side, developed using various Web technology (e.g. HTML/Java Script and Java Applet). It can display the information requested by the users for different purposes, e.g. user initialises the baseline technical process, launch the negotiation process when facing a conflicting issue, review the negotiation result after getting to an agreement.
- The controller layer implements the business logic in order to manage the data modules (Stakeholder, Process, and Negotiation). The process management module manipulates design process and models the tasks that the stakeholders work on. The negotiation management module helps stakeholders plan, enact and complete a negotiation process based on the argument-based approach. It also can help design team obtain the agreement of the negotiation process using argument evaluation functions. The stakeholder management module administrates the user account, background, preference, skills and other information.
- The model layer is responsible for accessing and manipulating the data for different objects, including processes, users, and negotiations.

Based on the functionalities specification and architecture, a Web based negotiation process management system is built and utilized for the case study in this research project. In this prototype system, users can register their own profiles, build software design concept structure, declare individual perspectives, generate argument structure based on concepts and perspectives, exchanging arguments with each other using Web technology, view related negotiation information collected by the system, and apply argument evaluation techniques for negotiations.

5.2 IWANT Prototype Implementation

A prototype of IWANT is being implemented in Java language, and is being deployed to support a few software

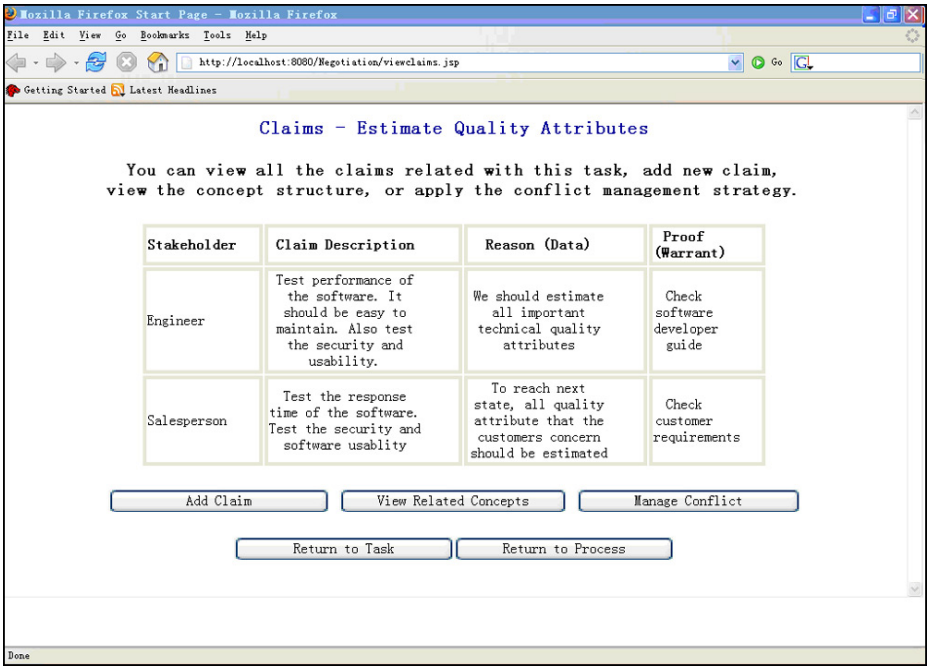

Figure 8. A snapshot of argument claims

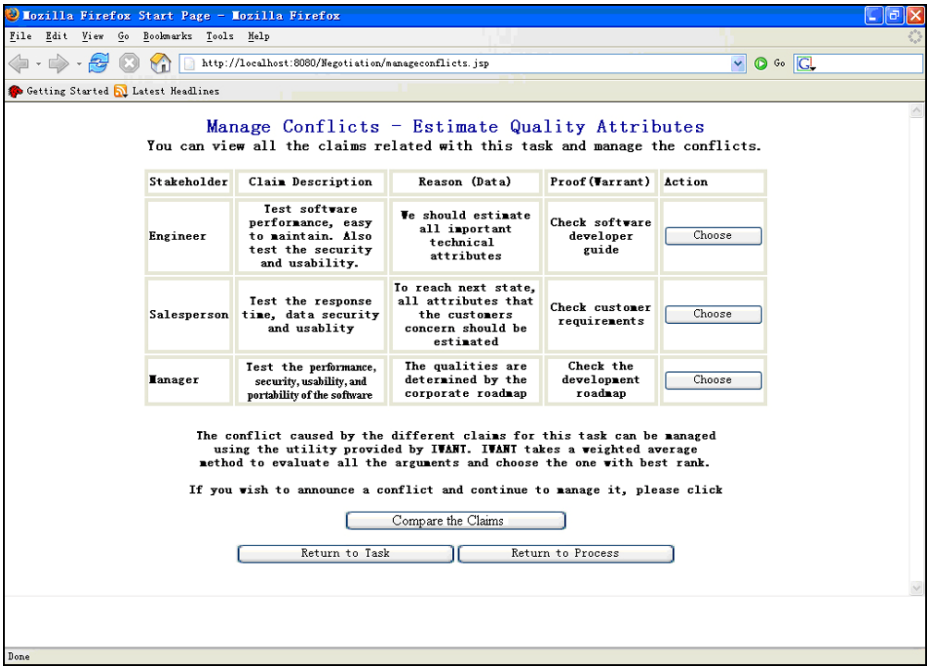

Figure 9. A snapshot of conflict management

development projects. The prototype is implemented as a web service on the Apache web server so that the stakeholders (users) can access this system via the Internet (Fielding *et al*., 1997, Apache Software Foundation 2006). To better illustrate the use of this prototype, three screen snapshots are taken in the prototype and provided in the figures below. Figure 8 is an example of showing the claims related with one task, which the stakeholders have different opinions of. The stakeholders can add new claim (e. g. the manager can add his claim following those made by the engineer and salesperson), view the concept structure (related with the conflicting task), or enter the conflict management phase. In the claims table, more user-friendly terms have been implemented, for instance, reason (as data in argument structure) and proof (as warrant), to improve the user experience.

Figure 9 presents the conflict management interface. The stakeholders review all the relevant claims and then either agree on one claim or use the system to rank all the claims by a weighted average method. Figure 10 shows the ranking result in an ordered list.

5.3 Initial IWANT Applications

IWANT is being used by a few software development groups to increase software design efficiency and validate the ASTN approach. Our plan is to integrate IWANT into their software design life cycle to specifically serve the negotiation process. We also use IWANT to compute negotiation efficiency and effectiveness statistics for future design team reference. These application experiments are being conducted in both academic units and software companies. A variety of user groups have been investigated and different social and technical factors have been collected and analyzed.

In these case studies, the following criteria were used to select this scenario for our case study:

1. The software design team is composed of multiple stakeholders, preferably with various backgrounds and domain knowledge, e.g. at least three distinctive disciplinary areas.
2. The design solutions call for collaborative negotiations via some systematic and iterative processes among stakeholders.
3. Each stakeholder is able to explain the required domain knowledge and their perspectives in a more or less structured manner. (Lu *et al*., 2005)

There are two main challenges in this practice that we are handling: first, due to the nature of software industry, i.e. high development cost and urgent marketing needs, the software design phase need to be short so that it can fit to the overall schedule for the business requirements. On the other hand, all stakeholders need to be inspired and guided to express own ideas and understand others' opinions accurately so the meaning of each concept and perspective can be well understood during the interaction.

According to the ASTN process, our case studies are taken in three phases: the pre-negotiation phase, the negotiation phase, and the post-negotiation phase. In the pre-negotiation phase, the stakeholders in our experiment will initiate a software design process (which consists of a series of technical design tasks) to develop a software

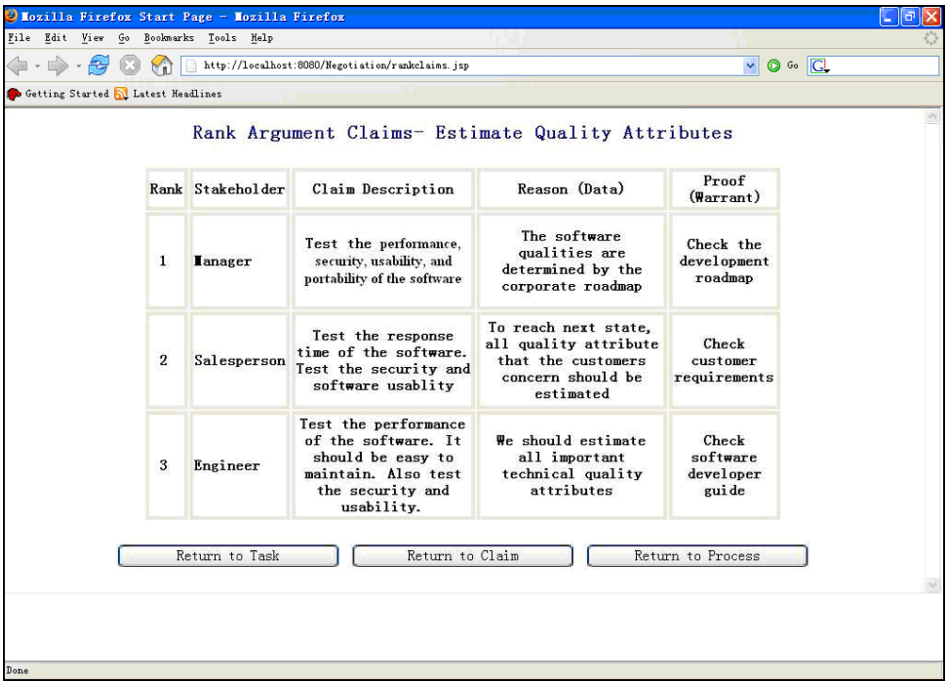

Figure 10. A snapshot of ranking argument claims

design solution. Either the design team will take their previously used design process or we will suggest a few in which the team may choose according to time, cost and other factors. The stakeholders then work on each design task and the differences between their implementations is captured and lead the team to negotiation phase.

During the negotiation phase, the concept structure is jointly proposed by the stakeholders to express their understanding (e.g. objective and criteria) about the task (in which the conflict occurs) and the perspective model of each stakeholder are captured to indicate their support or disagreement. Then our approach will use all the information to generate arguments for stakeholders and guide them through the argument-based negotiation process. At the end of this process, we will engage the argument evaluation approach to choose a most desired claim as an agreement for the conflicting issue for the team if necessary.

In the post-negotiation phase, the design team should already have a software design commonly agreed and accepted by all involved stakeholders. The shared concepts and common understood perspectives will be assets for future collaboration of this team. Our continuing work in this phase will be to summarize the stakeholder negotiation activities and evaluate the collaboration quality in terms of productivity and cost efficiency in the software design process. These statistics will also be provided to the management of this design team for future task-work refinement.

6. CONCLUSION AND FUTURE WORK

This paper presents a new approach to manage integrated socio-technical negotiation activities in a collaborative software design process. We have investigated the critical issues of such collaborative negotiation activities, including modeling negotiation arguments based on social and technical factors and analyzing these arguments to reconcile the conflicts for software design tasks. To address these challenges, we have developed a new approach based on the integration between the Socio-Technical Co-construction Process (STCP) and the Argument-based Negotiation Process (ABNP).

This research is based on a belief that software design is not only a technical decision making process conducted by a group of software experts, but also a social interaction process among all of the interested participating stakeholders. Based on this more comprehensive view, this research approach removes some of the critical limitations of traditional software design, such as providing a social interaction model to trace the source of the decision conflicts and clearly specifying an argument-based negotiation process to resolve the conflicts in collaborative design.

Additionally, this new approach takes advantage of an argument-based negotiation process that assists stakeholders in generating and evaluating their argument claims systematically based on their technical knowledge and social interaction. The identified conflicts among the stakeholders are systematically handled by the managed negotiation activities and the software design process is thus much improved. A research software prototype IWANT is being built to validate the proposed work and evaluated in several real-life software development projects. Then it will collect experimental data and user feedback both of which will then be used to summarize conflict profiles, calculate negotiation efficiency, and measure collaboration quality.

In conclusion, the proposed research is expected to provide a more comprehensive yet practical method for software design team to manage conflict negotiation and develop a shared software design solution. It gains us a deeper understanding regarding how to manage social interactions and their relations to technical decisions that occur in many real-life software design tasks. We also wish to transfer the lessons learned to other fields of engineering designs, such as new product developments, to broaden the research impacts. Our future research work will refine the conflict management strategies by defining design conflict profiles and their relationships with design tasks and stakeholders' perspectives. Furthermore, we plan to thoroughly validate this research framework and exercise the software prototype by conducting more case studies with the software industry. When more developments are conducted and application results are gathered, the framework and system will be continuously improved to eventually leading to the establishment of a scientific foundation for collaborative engineering.

ACKNOWLEDGEMENTS

We are grateful for the continual support provided by the U.S. Army Construction Engineering Research Laboratories (CERL) and the National Science Foundation. We thank Drs. Michael Case and Francois Grobler of CERL for their technical contributions. Special thanks are given to Mr. Andrew Wachter for his valuable technical editing and advice. We thank all of our colleagues in the IMPACT Laboratory at University of Southern California for reviewing this work and for joining our discussion of this work.

REFERENCES

Amgoud, L., Maudet, N., and Parsons, S. (2000), Modelling dialogues using argumentation, *MultiAgent Systems, Proceedings of Fourth International Conference*, **10-12**, 31-38.

Aldrich J., Garlan D., Schmerl B., Shaw M., and Wing J. (2006), Software engineering research in the computer science department of at Carnegie Mellon University, a web tutorial,

http://www.csd.cs.cmu.edu/research/areas/softeng/.

Alur, D., Crupi J., and Malks D. (2001), *Core J2EE Patterns: Best Practices and Design Strategies*, Prentice-Hall. ISBN 0130648841.

Anderson R. M., Hobbs B. F., and Bell M. L. (2002), Multiobjective decision making in negotiation and conflict resolution. Chapter 6, *Formal Models for Conflict Resolution*, edited by K.W. Hipel, prepared for *The Encyclopedia of Life Support Systems (EOLSS)*.

Arrow K. J. (1951), Social choice and individual values, Wiley, New York.

Avery J.,Yearwood J., and Stranieri A. (2001), An argumentation based multi-agent system for eTourism dialogue, *Proceedings of International Workshop on Hybrid Intelligent Systems (HIS '01)*, December 2001, Adelaide, Australia, **12**,194-210.

Berger P. L. and Luckmann T. (1967), *The Social Construction of Reality*, Anchor Books, 1967.

Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J., and Madachy, R. (1999), A stakeholder win-win approach to Software Engineering Education, *Annals of Software Engineering*.

Boehm B., Port D., Huang L. G., and Brown W. (2002), Using the spiral model and MBASE to generate new acquisition process models: SAIV, CAIV, and SCQAIV, *CrossTalk*, 20-25.

Buckingham Shum, S., MacLean, A., Bellotti, V. and Hammond, N. (1997), Graphical argumentation and design cognition, *Human-Computer Interaction*, **12** (3), 267-300.

Chang A. M. and Han T. D. (1995), Design of an argumentation-based negotiation support system, System Sciences, 1995. Vol. IV. P*roceedings of the Twenty-Eighth Hawaii International Conference*, 4(3-6), 242 -251.

Etzioni A. (1967), Mixed-Scanning: a third approach to decision-making, *Public Administration Review*, **27** (5), 385-392.

Fielding, R.T. and Kaiser, G. (2006), The Apache HTTP server project", *Internet Computing, IEEE*, **1** (4), 88-90.

Hart, K. A. (1990), Teaching thinking in college, *Accent on improving college teaching and learning* (ERIC Document Reproduction service No. ED 332 613).

Houp, K. W., Pearsall T. E., and Tebeaux E. (1998), *Reporting Technical Information,* 9th Edition. Oxford UP New York. 1998.

In, H., Olson D., and Rodgers T. (2002), Multi-criteria preference analysis for systematic requirements negotiation, *IEEE International Computer Software and Applications Conference* (COMPSAC 2002), 887-892, Oxford, UK.

Janssen, T. and Sage, A. P.(1996), Group decision support using Toulmin argument structures, *IEEE International Conference on Systems, Man, and Cybernetics*, **4**, 2704-2709 .

Jin Y., Geslin M., and Lu S. C.-Y. (2005), Impact of argumentative negotiation on collaborative engineering, IMPACT Laboratory, Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, USA.

Kazman, R. (2005), The essential components of software architecture design and analysis, *Proceedings of Software Engineering Conference,* Dec. 2005 Page(s): 1 pp. Digital Object Identifier 10.1109/APSEC.2005.103.

Kraus S. (2001), Automated negotiation and decision making in multiagent environments, Lecture Notes in Artificial Intelligence 2086, 150.

Lagomasino, A. and Sage, A. P. (1985), Representation and interpretation of information for decision support with imperfect knowledge, *Large Scale Systems*, 9(2), 169-181.

Lagomasino, A. and Sage, A. P. (1985), An interactive inquiry system, Large Scale Systems, **9**(3), 231-244.

Laskey, K. B., Chen, M. S., and Martin, A. W. (1989), Representing and eliciting knowledge about uncertain evidence and its implications, *IEEE Transactions on Systems, Man, and Cybernetics*, **19**(3), 536-545.

Lee Y., Choi H.-J. (2005), Experience of combing qualitative and quantitative analysis methods for evaluating software architecture, *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science* (ICIS 2005), 152-157.

Lu S. C-Y.and Cai J. (2001), "A collaborative design process model in the sociotechnical engineering design framework, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**, 3-20.

Lu S. C.-Y., Zhang J.-Y., Wang C-T., and Grobler F. (2005), Modelling design processes and stakeholder perspectives to support collaborative engineering negotiation: a case study of designing individualised prostheses over the internet, *International Journal of Computer Applications in Technology*, **23**(1), 2-12.

Marttunen, M. (1992), Commenting on written arguments as a part of argumentation skills-comparison between students engaged in traditional vs on-line study, *Scandinavian Journal of Educational Research*, **36**(4), 289-302.

Moore, M., Kazman R., Klein, M., and Asundi J. (2003), Quantifying the value of architecture design decisions: lessons from the field, *Proceedings of the 25th International Conference on Software Engineering (ICSE 25)*, Portland, Oregon, May 2003.

Nemhauser, G. L., Rinnoy Kan, A. H. G. and Todd, M. J. (1989), *Handbooks in Operations Research and Management Science*, **1**, Optimization, North-Holland, Amsterdam.

Rong J., Geng S. J., Valasek, J., Ioerger, T. R. (2002), Air

traffic conflict negotiation and resolution using an onboard multi-agent system, *Proceedings of Digital Avionics Systems Conference*, The 21st, **2**, 7B2-1-7B2-12.

Saaty T. L. (1980), *The Analytic Hierarchy Process,* New York: McGraw-Hill.

Sage, A. P. (1991), On the processing of imperfect information using structured frameworks, Chapter 7 in Kandel, A. (Ed.), *Fuzzy Expert Systems*, CRC Press, New York, 99-1 12.

Sage, A. P. (1992), *Systems Engineering*, John Wiley and Sons, New York.

Schum, D. (1994), *Evidential Foundations of Probabilistic Reasoning*, John Wiley and Sons, New York.

Sierra, C., Jennings, N. R., Noriega, P., and Parsons, S. (1998), A framework for argumentation-based negotiation, Intelligent Agent IV, International Workshop on Agent Theories, Architectures and Languages (ATAL-1997) (*Lecture Notes in Aritificial Intelligence*), Berlin, Springer-Verlag, **1365**, 177-192.

Sillince J. A. A. and Saeedi M. H. (1999), Computer-mediated communication: problems and potentials of argumentation support systems, *Decision Support Systems*, **26**, 287-306.

Simon H. (1957), A behavioral model of rational choice, in Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting, New York, Wiley.

Smith, D. G. (1977), College classroom interactions and critical thinking, *Journal of Educational Psychology*, **69**(2), 180-190.

Steuer, R. E. (1986), *Multiple Criteria Optimization*: *Theory, Computation and Application*, Wiley Series in Probability and Mathematical Statistics, John Wiley, New York, 546.

Toulmin, S. (1958), *The uses of argument*, Cambridge University Press, London.

Toulmin S., Rieke R., and Janik A. (1984), *An Introduction to Reasoning, Macmillan Publishing*, New York.

Yoon, K. P. and C.-L. Hwang (1995), *Multiple Attribute Decision Making*: *An Introduction*, Thousand Oaks, Sage Publications.

Wall, J. A., Calister, R. R. (1995), Conflict and its management, *Journal of Management*, **21**(2), 515-558.