A socio-technical negotiation approach for collaborative design in software engineering

Stephen C-Y. Lu and Nan Jing*

The IMPACT Research Laboratory,
Viterbi School of Engineering,
University of Southern California,
Los Angeles, CA 90089, USA
E-mail: sclu@usc.edu
E-mail: jing@usc.edu
*Corresponding author

Abstract: To support collaborative design in software engineering, we have built a socio-technical negotiation approach by integrating a Socio-Technical Co-construction Process (STCP) with an Argument-Based Negotiation Process (ABNP). The STCP provides rich contextual information of technical decisions and social interactions in a software design process. The ABNP provides STCP with a conflict resolution strategy by guiding software engineers to generate, exchange and evaluate their argument claims in negotiation activities. This paper reviews relevant research work and presents each step of this negotiation approach. 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 the enhanced negotiation capabilities in a dynamic socio-technical framework.

Keywords: collaborative engineering; socio-technical framework; ECN; engineering collaboration via negotiation; collaborative software design.

Reference to this paper should be made as follows: Lu, S.C-Y. and Jing, N. (2009) 'A socio-technical negotiation approach for collaborative design in software engineering', *Int. J. Collaborative Engineering*, Vol. 1, Nos. 1/2, pp.185–209.

Biographical notes: Stephen C-Y. Lu is permanent holder of the David Packard Chair in Manufacturing Engineering at University of Southern California (USC). He Heads the Master of Science in Product Development Engineering program at USC. He is the founding Director of the Improving Productivity via Advanced Collaboration Technology (IMPACT) research laboratory at USC. As a pioneer of the Engineering Collaboration via Negotiation (ECN) paradigm, he has published over 300 technical papers and served on many journals' editorial boards and keynote speaker in the fields of collaborative engineering. He is the Chief Editor of the *International Journal of Collaborative Engineering*. His current research interests are in design thinking, collaborative engineering, and technological innovation.

Nan Jing is a senior software engineer in Antenna Software, where he is in charge of the technical efforts for mobility platform such as blackberry, for business solutions including customer relationship management and inter-enterprise workflow modelling. He has published papers on the topics of software design, collaborative negotiation and business process management.

He co-authored a book in web service and Java technology, published by TsingHua University Press, China in 2003. He is a frequent reviewer for conferences and journals in information system and mobile technologies, and he is presently on the editorial review board of *International Journal of Handheld Computing Research*. He has earned a PhD Degree from the University of Southern California and a Bachelor Degree from Peking University, China, both in Computer Science.

1 Introduction

Software engineering 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, analyse, implement and maintain software solutions that can satisfy complex and evolving customer requirements. Nowadays driven by industry globalisation and internet revolution, most software design is carried out by distributed teams that include developers, architects and managers who have varying backgrounds and expertise. Therefore, a sound collaborative design methodology is needed for modern software engineering; the challenge of developing such a methodology has been the theme of our research. This paper presents how a systematic negotiation methodology can be devised to support multiple software design stakeholders in making technical decisions collaboratively, in light of their social interactions with divergent backgrounds and skill sets, in addition to limited time and resources.

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). In real-life software design processes, software engineers always need to negotiate with each other in order to reach agreements 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., other non-technical factors) is just as important as the ability to develop computation algorithms and build data structures. In previous software engineering practices, little attention was placed on these collaborative activities of software design, let alone the systematic supports to negotiation tasks. For a multi-disciplined software design team, collaborative negotiation is an important and indispensable task that should be fully understood and systematically practiced by all those involved.

This paper focuses on the collaborative negotiation tasks in which a team of software engineers must work with each other to design a software solution. The subject of collaborative negotiation in engineering is part of an emerging research field, called collaborative engineering (Lu, 2003). 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* endeavour 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. In our research, 'social' refers to the behaviours that take the interests of others into account and the cooperative characteristics between individuals. We also use the word 'social' to represent the common stakeholder characteristics, which influence collaborative team dynamics during social interactions. These characteristics include mostly the non-technical aspects of an individual stakeholder, such as background, objective, interest and criteria. 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 summary, the above definitions in our research explicitly acknowledge collaborative engineering 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 an engineering team.

Specifically, this paper presents a new socio-technical approach to support collaborative negotiation tasks by modelling, tracking and managing stakeholders' negotiation arguments, which continuously evolve during the collaborative process of software design. This new approach helps stakeholders to organise and generate argument claims in preparation for systematic negotiations, as well as helping reconciles design conflicts by recommending potential conflict management strategies. The conflict management strategies include, for example, how to evaluate possible alternatives from these negotiation arguments and compare these alternatives in order to choose desirable ones in the applicable circumstances.

Section 2 reviews some related past works developed through software engineering research in this field, including the Architecture Trade-off Analysis Method (ATAM) approach, the Cost Benefit Analysis Method (CBAM) approach, the Multi-Criteria Preference Analysis Requirements Negotiation (MPARN) model, and a Win-Win approach. In Section 3, we present an Argument-based Socio-Technical Negotiation (ASTN) approach for collaborative design in software engineering. This new approach helps the design team systematically generate their negotiation argument claims based on both social (i.e., teamwork based on social interactions) and technical (i.e., task-work based on domain knowledge) factors. It also specifies how these argument claims can be exchanged among team members and evaluated to systematically complete the negotiation process. In Section 4 we describe a software application called the Intelligent Web-based Argument Negotiation Toolkit (IWANT), which is being developed and used to validate our research framework. Lastly, Section 5 summarises the lessons learned from this study and describes our planned future work.

2 Related work

Most of the existing negotiation research for collaborative design in software engineering falls in two categories: software design evaluation approaches and software design negotiation models. 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 (Lee and Choi, 2005). It helps the engineers understand the consequence of software design with respect to the system's quality attributed 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 (Kazman, 1998, 2005; Moore, 2003). 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, analyses, and makes technical decisions regarding software architecture design alternatives with consideration of economic factors (In 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 in the second category is the Win-Win negotiation model 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 model, 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 model stakeholders still need to generate and negotiate the architecture alternatives manually by a rather ad-hoc process (In et al., 2001, 2002). Furthermore, the Win-Win negotiation model is based on a software engineering approach called Model-Based Architecting and Software Engineering (MBASE), 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 modelled 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 present study is the MPARN model, which guides stakeholders from design options to agreements by using multi-criteria preference analysis techniques. 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 addressed 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 analyse and prioritise their design decisions, but does not specify a negotiation approach for the stakeholders to jointly achieve a common agreement.

Our research focuses on the specific task of socio-technical negotiations in collaborative design of software systems. The novelty of our approach is in providing guidance for the stakeholders to systematically generate, exchange, and evaluate their argument claims made during the negotiation process based on *both technical decisions and social interactions*. It integrates a baseline software design process with a conflict-reconciling process. As well, it explains to stakeholders how they can extract and generate relevant information from design tasks, and how this information can be used to resolve conflicts in negotiation. The remainder of this article describes this approach in details, and presents a software prototype system which implements the approach using internet-based computer technology.

3 Our approach

In the collaborative software design process, a team of stakeholders with different social backgrounds and technical expertise must jointly undertake many common tasks, |which require making joint decisions based on communal 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, we integrate a STCP framework (Lu, 2001) and an ABNP model (Toulmin 1958; Jennings, 1998) to build a socio-technical negotiation approach for the 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, applying STCP for the collaborative software design process overcomes the limitations of traditional work by explicitly modelling the social interactions and investigating both the social and technical factors. Meanwhile, the ABNP facilitates the conflict management in the STCP by systematically guiding the team through a negotiation process in which their differences in technical decisions are reconciled.

To address the above challenge, we integrate a Socio-Technical Co-construction Process (STCP) framework (Lu, 1999) and an Argument-based Negotiation Process (ABNP) model (Toulmin, 1958; Chang et al., 1995; Jennings, 1998; Sillince et al., 1999; Amgoud et al., 2000; Avery et al., 2001; Kraus, 2001; Rong et al., 2002) to build a socio-technical negotiation approach for the collaborative design process in software engineering. In the definition of the STCP, the co-construction process is one in which two or more individuals act cooperatively to jointly and dynamically construct each other's 'perspectives' toward a shared task to produce a common solution (or a shared reality), such as a design, a process, software, or a product. A 'perspective' is defined in our research as the particular ways (i.e., viewpoints) via which the stakeholder views the world and makes decisions. STCP builds a co-construction model for the collaborative engineering process using the following seven steps (see Figure 1):

1 First it defines a starting 'baseline process' for the chosen application domain (i.e., software design, in this case). 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 (SOPs) instituted by the company.

2 Secondly, STCP identifies a set of 'stakeholders' who have an interest in the outcomes of, and will directly or indirectly participate in, the co-construction process.

3 In Step 3, stakeholders propose an initial 'concept structure' to represent relevant social factors (e.g., background, objective, etc) which influence the collaborative processes and establish their initial 'perspective models' for each proposed concept.

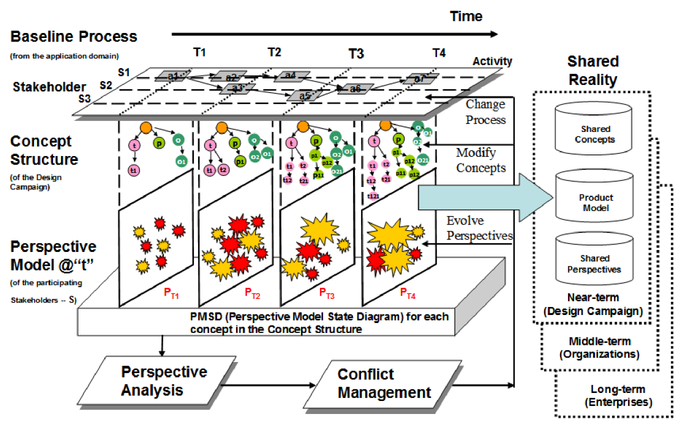
4 In Step 4, STCP performs analyses of these initial perspective models and manages the conflicts identified from these perspective models.

5 Based on these analysis results, relevant conflict management strategies in Step 5 of STCP can suggest changing the stakeholders' perspectives, concepts in the CS, and/or steps in the baseline process.

6 After these suggestions are implemented, the Step 6 of STCP begins another round of the co-construction process with the new (updated) stakeholders' perspectives until no further conflicts are detected from perspective analyses.

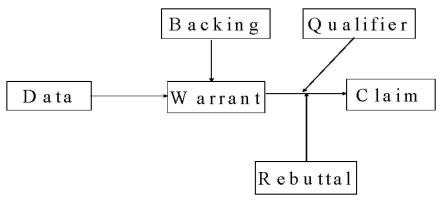
7 At the end (the Step 7), STCP obtains a 'shared reality' as a result of the co-construction process. Shared reality can include, for example, an agreed upon product model (in the case of product design), as well as a set of co-constructed concepts and stakeholders' perspectives, which are useful for similar design tasks in the future.

Figure 1 The Socio-Technical Co-Construction Process (STCP) (see online version for colours)



While providing a model for the co-construction process in collaborative design, STCP does not prescribe a specific conflict resolution framework for stakeholders to resolve conflicting decisions. Therefore, we use the ABNP model in our research to complement STCP by guiding the design team to generate, exchange, and evaluate negotiation argument claims during the collaborative co-construction process. The 'argument' in the ABNP framework is built based on the Toulmin's structure of argument (Toulmin, 1958), which provides the language symbols and a data structure that supports the argumentation process. Figure 2 illustrates the details of this structure. As shown in the figure, the Toulmin structure is mostly procedural, and its layout focuses on the movement of accepted *data* to the *claim* through a *warrant (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. Toulmin also recognises 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.

Figure 2 Toulmin's argument structure in ABNP



Note that ABNP, by itself, does not specify how to obtain the information for each component in its arguments. STCP, on the other hand, can provide ABNP with this critical information based on both social and technical factors captured in the co-construction model. On the technical side, the baseline process and its design tasks model the stakeholders' decisions for the software design. On the social side, the concept structure helps the stakeholder declare their characteristics (e.g., background, objective, interests, 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 3 describes an example scenario of the integration between the ABNP and STCP. As shown in the figure, the technical factors in the STCP provide the ABNP 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) in ABNP. Based on these factors, in the step to manage the conflicts, the STCP calls the ABNP as a negotiation strategy for conflict resolution.

The two complementary frameworks work together to establish a new ASTN approach for collaborative software design. Figure 4 illustrates this integrated approach, which has three inter-related phases. First, the Pre-negotiation phase starts with the baseline process and determines whether to initiate a negotiation by identifying all conflicting implementations of a design. It identifies the stakeholders, starts a design process, asks each stakeholder to propose an implementation for specific tasks in the design process, and then checks the differences (i.e., conflicts) between proposed implementations. Second, the Negotiation phase helps the stakeholders prepare for their arguments, and guide them into resolving the conflicts by an argument-based process. In this phase the stakeholders jointly propose a concept structure 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, which are then compared using an argument evaluation method. 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. The following sub-sections explain these three ASTN phases in more details.

Figure 3 Integration of STCP and ABNP (see online version for colours)

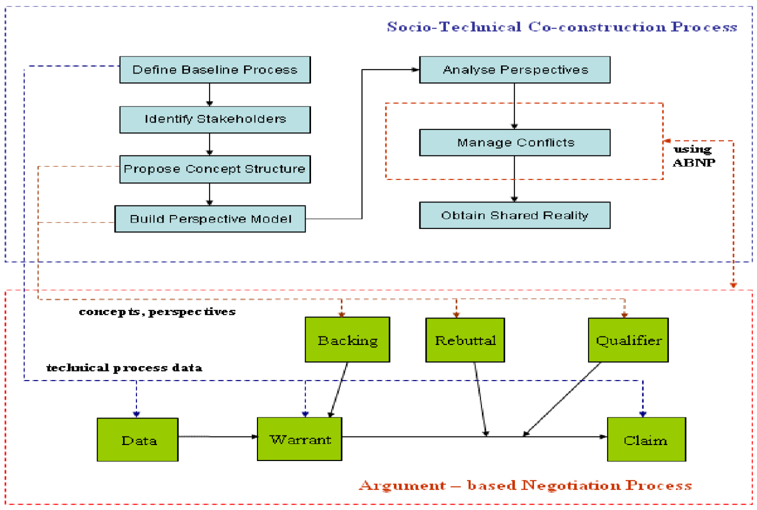
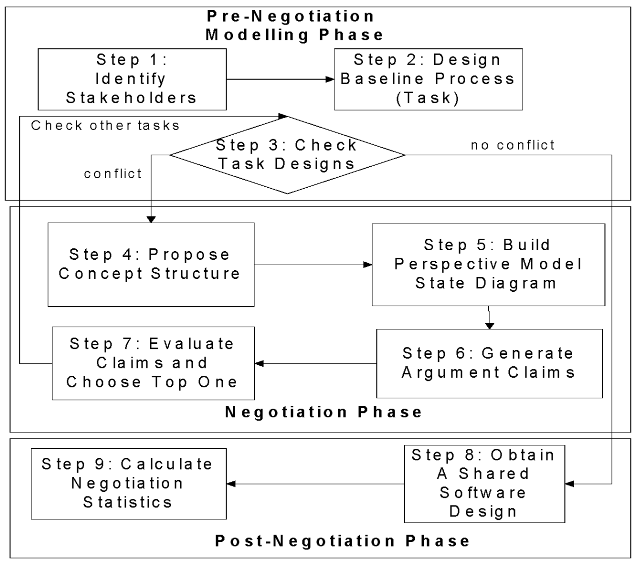


Figure 4 A negotiation approach for collaborative design in software engineering


Pre-Negotiation Modelling Phase
Step 1: Identify Stakeholders
Step 2: Design Baseline Process (Task)
Check other tasks
Step 3: Check Task Designs
no conflict
conflict
Step 4: Propose Concept Structure
Step 5: Build Perspective Model State Diagram
Step 7: Evaluate Claims and Choose Top One
Step 6: Generate Argument Claims
Negotiation Phase
Step 9: Calculate Negotiation Statistics
Step 8: Obtain A Shared Software Design
Post-Negotiation Phase

3.1 The pre-negotiation phase

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 organise 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 modelled 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.

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 3.3) with supporting agreements on how to implement all design tasks.

3.2 The negotiation phase

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 ABNP is undertaken according to the following two stages:

- stakeholders generate argument claims (or counter proposals) for concerned issues and provide supporting data
- 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). 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 rooted 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, for example, the objectives for which the task is undertaken, the criteria to make the implementation, and the alternatives, if available, to implement a different task to achieve the same objective. To better capture these differences and get a deeper understanding of the conflict, the ASTN approach provides a structure, which models the concepts that underline the proposed technical decisions. This concept structure is a model to organise 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 describes this example concept structure, including information about stakeholders, objectives, criteria, and alternatives. There are three stakeholders in this example: salesperson, engineer, and manager. Salesperson's objectives are to guarantee performance, security and usability of the software. Her criteria are that sale is most important and hence every attribute should be evaluated by sale requirements. The engineer, on the other hand, believes that software performance, maintainability, security, and usability are most important 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. The engineer and manager have provided two alternatives as different implementations for the design task.

Table 1 An example concept structure for 'estimate quality attributes'

Stakeholder	*Objectives*	*Criteria*	*Alternative*
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	n/a
Engineer	Performance, easy-to-maintain, security, and usability	The quality attributes should be determined based on appropriate development resource	Build a prototype to get software quality statistics
Manager	Performance, security, usability, and portability	Project responsibility and executive decisions	Import external software verification program

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 inter-related psychological and organisational factors. There is no practical way that a complete modelling 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 modelling 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 analysed next to systematically reconcile the conflicts in our negotiation approach.

Although stakeholder perspectives are often highly subjective in nature, some quantitative methods are needed in order to further analyse 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 1–5 scale where the ranking is as follow:

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, Tables 2–7 show some example perspectives of the stakeholders for the example design task 'Estimate Quality Attributes'. Tables 2 and 3 show the perspective models for their personal concepts. Tables 4–7 show the perspective models for the concepts of others. Positive numbers indicate support, negative indicate disagreement.

Table 2 Perspective model – stakeholders' desire for own objectives

Stakeholder	*Objectives*	*Desire* (*1–10*)
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' objectives

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

Table 4 Perspective model – stakeholders' desire for own criteria

Stakeholder	*Criteria*	*Desire* (*1–10*)
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 alternative

Stakeholder	*Alternative*	*Desire* (*1–10*)
Salesperson	n/a	n/a
Engineer	Software prototype	Software prototype: 4
Manager	External program	External program: 4

Table 7 Perspective model – stakeholders' support or disagreement for others' alternative

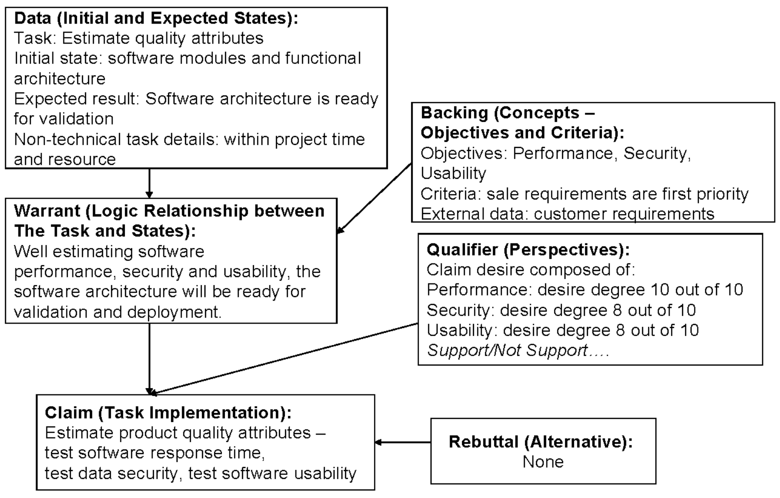
Stakeholder	*Sales*	*Engineers*	*Manager*
Salesperson	n/a	Software prototype: 4	External program: 4
Engineer	n/a	n/a	External program: 4
Manager	n/a	Software prototype: 3	n/a

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 representing 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 the alternatives to implement the tasks, used as rebuttal in the argument, provide other options (thus suspending the warrant). 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 underline reasons and desires (e.g., perspectives). Figure 5 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., his desire for the performance, security and usability. The qualifier also includes her support or disagreement on other's concepts.

Figure 5 An example argument (by the salesperson)


Data (Initial and Expected States):
Task: Estimate quality attributes
Initial state: software modules and functional architecture
Expected result: Software architecture is ready for validation
Non-technical task details: within project time and resource
Backing (Concepts – Objectives and Criteria):
Objectives: Performance, Security, Usability
Criteria: sale requirements are first priority
External data: customer requirements
Warrant (Logic Relationship between The Task and States):
Well estimating software performance, security and usability, the software architecture will be ready for validation and deployment.
Qualifier (Perspectives):
Claim desire composed of:
Performance: desire degree 10 out of 10
Security: desire degree 8 out of 10
Usability: desire degree 8 out of 10
Support/Not Support….
Claim (Task Implementation):
Estimate product quality attributes – test software response time, test data security, test software usability
Rebuttal (Alternative):
None

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 paper, a simple example is provided, as an illustration, using 'weighted average' to evaluate the arguments based on concepts and perspectives, e.g., objectives, criteria, alternatives, support, dissent. 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, 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 objectives, criteria and alternatives. The mathematical formula of this method is as follows:

  Claim evaluation = average (objectives evaluations + criteria evaluations + alternative evaluations).

- The method to evaluate an objective is to add the sum of the desires for this objective and the support or disagreement from others. The evaluation of this objective is represented by the 'weight' (i.e., perspective) added by all the stakeholders. Its mathematical formula is as follows:

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

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 methods to evaluate the criteria and the alternatives are same as that for the objectives (see below).

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

$$\text{Alternative evaluations} = \sum(\text{alternative\_desire} + \sum_{\text{stakeholder}} \text{support/stakeholder\_number} + \sum_{\text{stakeholder}} \text{disagreement/stakeholder\_number}) / \sum \text{alternative\_number}.$$

For example, the evaluation for the manager's claim is:

$$\begin{aligned}\text{Claim evaluation} &= \text{average (objectives evaluations + criteria evaluations + alternative evaluations)}\\ &= [(\text{performance\_evaluation} + \text{security\_evaluation} + \text{usability\_evaluation} + \text{portability\_evaluation})\\ &\quad + (\text{project responsibility\_evaluation} + \text{executive power\_evaluation})\\ &\quad + (\text{external program\_evaluation})]/3\\ &= [[(5 + 5 + 5)/3 + (5 + 5 + 5)/3 + (4 + 5 + 5)/3 + (5 + 3 + 3)/3]/4\\ &\quad + [(5 + 4 + 4)/3 + (4 + 5 + 5)/3]/2 + (4 + 4 + 4)/3]/3\\ &= 4.33.\end{aligned}$$

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 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.

3.3 The post-negotiation phase

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, and have converged onto one common software design, the collaboration statistics are calculated and summarised. 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 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 management 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.

4 Research prototype

Using the ASTN approach presented in the previous sections, we have been developing an IWANT. IWANT is a computer-supported negotiation system based on the ASTN approach, which implements a socio-technical ABNP for the collaborative software design. The uniqueness of this system includes the modelling and analysis of the social interactions and technical decisions of the stakeholders. 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, architecture, implementation and application of the IWANT research prototype.

4.1 The functionality of IWANT

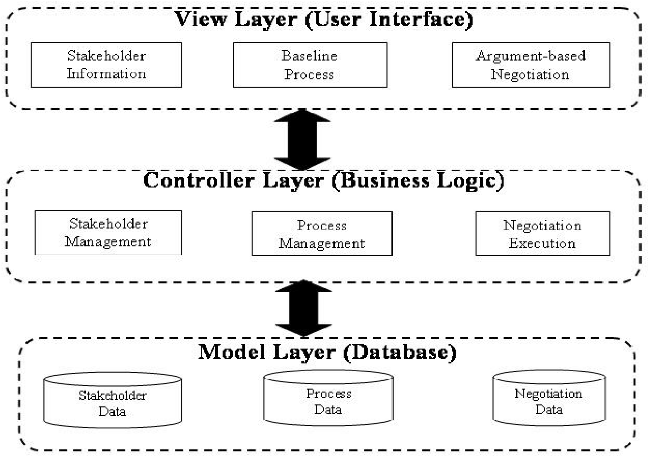
The major functionality of IWANT is to help stakeholders to systematically negotiate their design conflicts based on both technical and social factors. When stakeholders realise that there are different implementations in their software design tasks, they can activate the negotiation activity by logging into the IWANT system and starting 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 support 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.

4.2 IWANT system architecture design

Based on the specified functionalities, Figure 6 shows the overall system architecture of IWANT.

The architectural design of IWANT has three layers in accordance with the widely accepted Model-View-Controller standard (Sun Microsystems, 2002; Selfa et al., 2006) the *view layer* is the user interface component running on the client side, and displays the information requested by the users; the *controller layer* implements the business logic in order to process and modify the data accessed, and it is manipulated by the *model layer* with several data structures (Stakeholder, Process and Negotiation). Within the controller layer, the negotiation execution module helps stakeholders plan, enact and complete a negotiation process based on the argument-based approach. The process management module manipulates design process and models the tasks that the stakeholders work on. And the stakeholder management module manages the stakeholder account – background, preference, skills and other information.

Figure 6 The IWANT system architecture



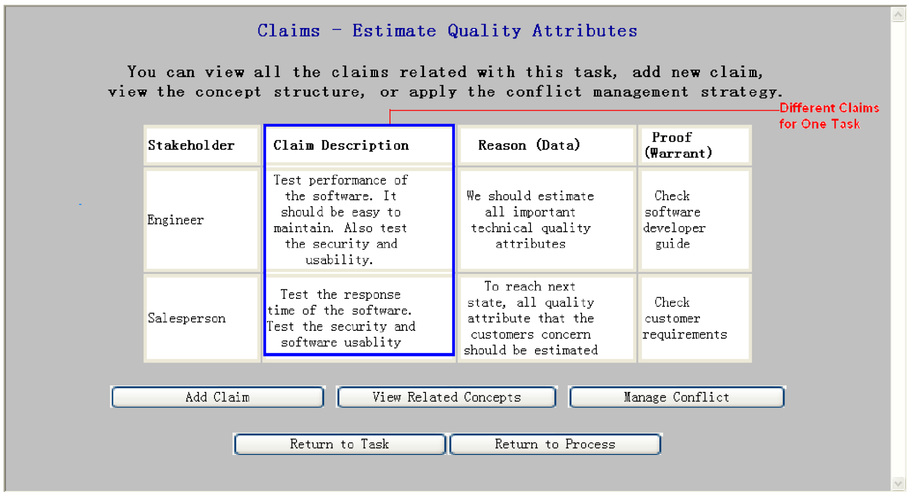
4.3 IWANT prototype implementation

A prototype of IWANT is being implemented in Java language, and is being deployed to support a few software 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. At the beginning, they will jointly implement each design task in the process management module, and their profiles will also be captured by the stakeholder management module (see Figure 6). The ABNP is implemented in the negotiation execution module and enforced by the system to define how a negotiation session is conducted. At any point in time during this process, stakeholders can propose a claim for a task and declare their concepts and perspectives. They can review the arguments claims of each other on IWANT and choose one to resolve the difference manually or by evaluation. A negotiation process ends with an agreement on the different claims when either of the following conditions is met:

- all the involved stakeholders agree with one claim before time is up
- when the allowed time has elapsed, an evaluation approach is applied and a claim with highest ranking is chosen.

To better illustrate the use of this prototype, four screen snapshots are taken in the prototype and provided in the figures below. Figure 7 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 8 illustrates a typical concept structure, which lists the concepts related with the conflicting task. The table in this page provides the information for each concept: the stakeholder who proposes this concept, the type of the concept (e.g., objective, criteria), and the description provided by the stakeholder. On this user interface, the stakeholders can declare their perspectives for each concept or add a new concept for the task.

Figure 7 A snapshot of argument claims (see online version for colours)

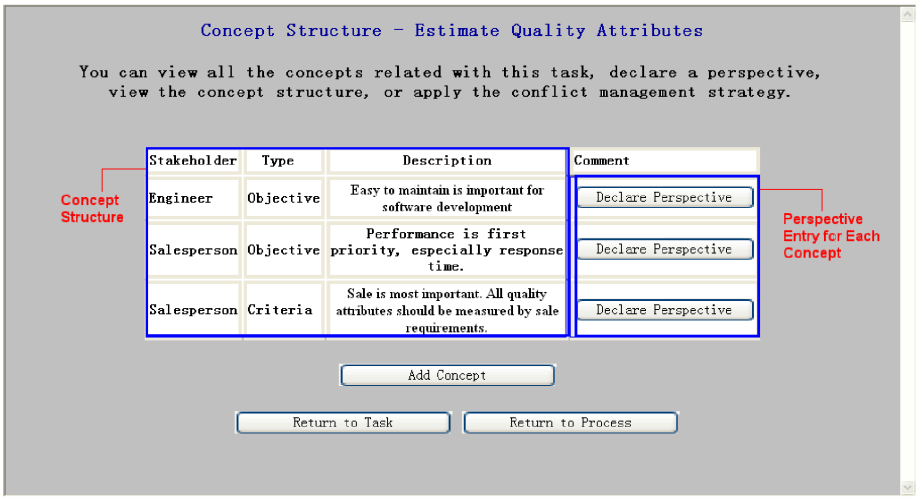


Claims – Estimate Quality Attributes

You can view all the claims related with this task, add new claim, view the concept structure, or apply the conflict management strategy.

Stakeholder	Claim Description	Reason (Data)	Proof (Warrant)
Engineer	Test performance of the software. It should be easy to maintain. Also test the security and usability.	We should estimate all important technical quality attributes	Check software developer guide
Salesperson	Test the response time of the software. Test the security and software usablity	To reach next state, all quality attribute that the customers concern should be estimated	Check customer requirements

Figure 8 A snapshot of concept structure (see online version for colours)



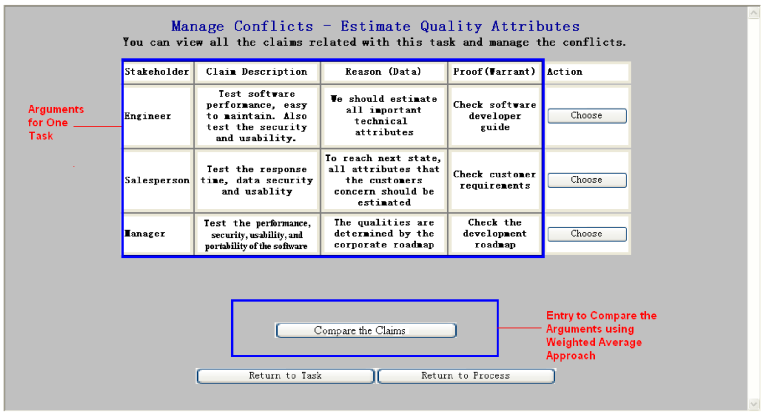
Concept Structure – Estimate Quality Attributes

You can view all the concepts related with this task, declare a perspective, view the concept structure, or apply the conflict management strategy.

Stakeholder	Type	Description	Comment
Engineer	Objective	Easy to maintain is important for software development	Declare Perspective
Salesperson	Objective	Performance is first priority, especially response time.	Declare Perspective
Salesperson	Criteria	Sale is most important. All quality attributes should be measured by sale requirements.	Declare Perspective

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.

Figure 9 A snapshot of conflict management (see online version for colours)

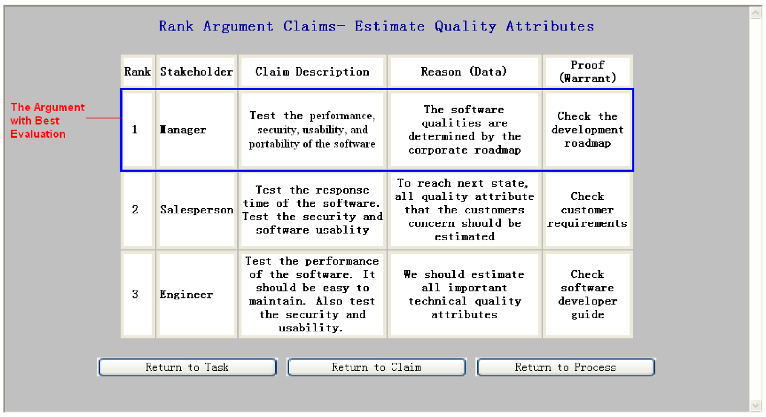


Manage Conflicts – Estimate Quality Attributes

You can view all the claims related with this task and manage the conflicts.

Stakeholder	Claim Description	Reason (Data)	Proof(Warrant)	Action
Engineer	Test software performance, easy to maintain. Also test the security and usability.	We should estimate all important technical attributes	Check software developer guide	Choose
Salesperson	Test the response time, data security and usablity	To reach next state, all attributes that the customers concern should be estimated	Check customer requirements	Choose
Manager	Test the performance, security, usability, and portability of the software	The qualities are determined by the corporate roadmap	Check the development roadmap	Choose

Figure 10 A snapshot of ranking argument claims (see online version for colours)



Rank Argument Claims– Estimate Quality Attributes

Rank	Stakeholder	Claim Description	Reason (Data)	Proof (Warrant)
1	Manager	Test the performance, security, usability, and portability of the software	The software qualities are determined by the corporate roadmap	Check the development roadmap
2	Salesperson	Test the response time of the software. Test the security and software usablity	To reach next state, all quality attribute that the customers concern should be estimated	Check customer requirements
3	Engineer	Test the performance of the software. It should be easy to maintain. Also test the security and usability.	We should estimate all important technical quality attributes	Check software developer guide

4.4 *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 analysed. A collaboration statistics template has also been developed and provided to the experiment units. Table 9 shows this example template. Using the template, IWANT is able to efficiently collect the complete data sets, including tasks, stakeholders' concept structures, stakeholders' perspectives, stakeholders' arguments and conflicts.

Table 9 Example collaboration statistics template

Tasks					
Task Name	Initial State	Expected State	Has A Conflict (Y/N)		
Stakeholders					
Stakeholder Name	Title	Task	Role	Task Implementation	
Concept structure					
Concept Name	Stakeholder Name	Type	Description	Task Name	
Perspectives					
Concept Name	Stakeholder Name	Perspective Value	Note		
Argument					
Stakeholder Name	Claim	Data Warrant	Backing	Rebuttal	Qualifier
Conflict					
Conflicting Task	Involved Stakeholders	Winner Claim	Time Usage	Conflict Description	
			Practical Expected		

In the ongoing experiments, an interesting issue in deploying the IWANT system was found: one experiment unit already had a software design system and hence the stakeholders were reluctant to manage the design process with IWANT. This situation has inspired us to improve IWANT by directly importing the software design process via a common data standard, such as XML. As such, the currently used software design system in the experiment unit can export the design process in XML. IWANT will then transform this XML data to its own format and render the process to the stakeholders. This system application also opens the direction of future investigations on how to gain the acceptance of the ASTN approach by user groups and how to better deploy IWANT if the stakeholders already have other software design systems in use.

5 Conclusion and future work

This paper presents a new approach to support integrated socio-technical negotiation activities in a collaborative software design process. We have investigated the critical issues of such collaborative negotiation activities, including modelling negotiation arguments based on social and technical factors and analyse 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 STCP and the ABNP.

We believe 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, our 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 a negotiation process to resolve the conflicts in collaborative design. Additionally, this new approach takes advantage of an ABNP 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 negotiation activities and the software design process is thus much improved. A software prototype called IWANT is being developed and evaluated in several real-life software development projects. User feedbacks and negotiation efficiency statistics are being collected to validate our research and improve the approach. In conclusion, our approach provides a more comprehensive yet practical method for stakeholders to negotiate conflicting opinions and develop a shared software design solution.

Our future research work will refine the conflict management strategies by defining design conflict profiles and their relationships with design tasks and stakeholders' perspectives. Also based on the collaboration statistics template, we will further propose a standard to evaluate the quality of collaboration according to negotiation efficiency and conflict profiles. In addition, we hope to gain a deeper understanding of social interactions and their relations to technical decisions that occur in many real-life software design tasks. Furthermore, we plan to thoroughly validate this research framework and exercise the software prototype by conducting more case studies with the software industry. We also wish to transfer the lessons learned to other fields of engineering designs, such as new product developments, to broaden our research impacts. 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 US Army Construction Engineering Research Laboratories (CERL) and the National Science Foundation. We thank Dr. 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 USC for reviewing this work, and for suggesting the use of Wiki media for internet based discussion.

References

Aldrich, J., Garlan, D., Schmerl, B., Shaw, M. and Wing, J. (2006) *Software Engineering Research* in the Computer Science Department of at Carnegie Mellon, A Web Tutorial, http://www.csd.cs.cmu.edu/research/areas/softeng/

Amgoud, L., Maudet, N. and Parsons, S. (2000) 'Modelling dialogues using argumentation, multiagent systems', *Proceedings of Fourth International Conference*, 10–12 July, pp.31–38.

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, Adelaide, Australia, Vol. 12, pp.194–210.

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*, pp.295–321.

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*, January, pp.20–25.

Chang A. M. and Han T. D. (1995), 'Design of an argumentation-based negotiation support system', *System Sciences, 1995. Vol. IV*, *Proceedings of the Twenty-Eighth Hawaii International Conference on Volume: 4*, 3–6 January, Vol. 4, pp.242–251.

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)*, Oxford, UK, pp.887–892.

In, H., Olson, D. and Rodgers, T. (2001) 'A requirements negotiation model based on multi-criteria analysis', *Requirements Engineering, Proceedings of Fifth IEEE International Symposium on 27–31 August*, pp.312, 313.

Kazman, R. (2005) 'The essential components of software architecture design and analysis', *Software Engineering Conference, 2005. APSEC '05. 12th Asia-Pacific 15–17 December 2005*, Page(s):1 pp. Digital Object Identifier 10.1109/APSEC.2005.103.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H. and Carriere, J. (1998) 'The architecture tradeoff analysis method', *Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on 10–14 August*, pp.68–78.

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

Lee, Y. and 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)*, pp.152–157.

Lu, S.C-Y. (2003) *Engineering as Collaborative Negotiation: A New Paradigm for Collaborative Engineering Research*, the ECN Working Group of the International Institution of Production Engineering Research (CIRP), see http://wisdom.usc.edu/ecn

Lu, S.C-Y. and Cai, J. (1999) 'Modeling collaborative design process with a socio-technical framework', *Proceedings of Sixth ISPE Int'l Conf. Concurrent Engineering*, Bath, UK.

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.

Rong, J., Geng, S.J., Valasek, J. and Ioerger, T.R. (2002) 'Air traffic conflict negotiation and resolution using an onboard multi-agent system', *Digital Avionics Systems Conference, 2002. Proceedings of the 21st, Volume: 2*, Vol. 2, pp.7B2-1–7B2-12.

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

Selfa, D.M., Carrillo, M. and Del Rocio Boone, M. (2006) 'A database and web application based on MVC architecture', *Electronics, Communications and Computers, CONIELECOMP 2006, 16th International Conference on 27-01 February*, pp.48–48.

Sierra, C., Jennings, N.R., Noriega, P. and Parsons, S. (1998) 'A framework for argumentation-based negotiation', in Singh, R.A. and Wooldridge, M. (Eds.): *Intelligent Agent IV: 4th, International Workshop on Agent Theories, Architectures and Languages (ATAL – 1997)*, Lecture Notes in Aritificial Intelligence, Springer-Verlag, Berlin, Vol. 1365, pp.177–192.

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

Sun Microsystems (2002) *Designing Enterprise Applications with the Java 2 EE Platform*, 2nd ed., http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/titlepage.html

Toulmin, S. (1958) *The Uses of Argument*, Cambridge University Press, London.