Model-based Analysis and Visualization of Conflicting Requirements in the Early Stages of Software Development

by

Kedar Shrikhande
B.Eng., University of Saskatchewan, 2005
B.Sc., University of Saskatchewan, 2005

A Thesis Submitted in Partial Fullfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Model-based Analysis and Visualization of Conflicting Requirements in the Early Stages of Software Development

by

Kedar Shrikhande
B.Eng., University of Saskatchewan, 2005
B.Sc., University of Saskatchewan, 2005

Supervisory Committee

Dr. Daniela Damian, (Department of Computer Science)
Supervisor

Dr. Alexandra Branzan Albu, (Department of Electrical Engineering)
Co-Supervisor

Dr. Yvonne Coady, (Department of Computer Science)
Departmental Member

Dr. Florin Diacu, (Department of Mathematics)
Outside Member

Supervisory Committee

Dr. Daniela Damian, (Department of Computer Science)
Supervisor

Dr. Alexandra Branzan Albu, (Department of Electrical Engineering)
Co-Supervisor

Dr. Yvonne Coady, (Department of Computer Science)
Departmental Member

Dr. Florin Diacu, (Department of Mathematics)
Outside Member

Abstract

Many of the failures and deficiencies of software projects can be attributed to the lack of effort exerted in addressing requirements in the early planning stages of software development. In multiple stakeholder development environments, requirements will inevitably come into conflict, therefore, it is important to address these conflicts early in software development.

The research presented in this thesis surveyed several existing models that resolve requirements conflicts. The goal of the research was to investigate the suitability of these models in identifying, visualizing and solving requirements conflicts. To achieve this goal it was decided to apply these models in a context different than they were originally applied. The context of this research was a process where two stakeholder groups negotiated the requirements of the particular software system. The application

of the models was done as a case study. Three models were studied, namely the Utility Curves Model, the Win-Win Model and the ***i**** Framework.

It was found that each model contributes uniquely to conflict resolution. We have documented strength and limitations for each model and have concluded that these three models should be used together in tandem. A hybrid model was constructed that was composed of the three models. The hybrid model leverages the strengths and addresses the limitations of the three individual models.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor Dr. Daniela Damian for supporting me, challenging me and believing in me.

I would like to thank my co-supervisor Dr. Alexandra Branzan Albu for her fresh insights.

I would also like to thank my graduate committee members Dr. Yvonne Coady and Dr. Florin Diacu.

I would especially want to thank Chris, Laura-Lee, Lisa and Gerod for their support and friendship.

I would also like to acknowledge Luis, Florian, Irwin, Lucas, Sabrina, Rafael and Thanh for their constructive feedback.

Dedication

To Sameer and My Beloved Ajji

Chapter 1

Introduction

In the multiple stakeholder software development environment of today, requirements inevitably come into conflict. According to Easterbrook[1] stakeholders requirements often come into conflict because stakeholders have divergent goals and perspectives. Requirements conflicts should be resolved in the early planning stages of software development, because Boehm[2] has demonstrated that resolving such conflicts in the later stages of software development are far more costly than addressing conflicts early in the process. Brooks[3] asserts that specifying the requirements of a system is the most difficult part of software development, because if the requirements were specified incorrectly the system could be crippled later. However, Easterbrook[4] affirms that not enough effort is exerted in the early planning stages to understand stakeholder requirements and to deal with conflicting requirements. As a result, Faulk[5] has

shown that most software projects are being delivered late, over budget, and/or of lower quality. Conflict resolution is worthwhile studying, because of the shortfalls in addressing requirements conflicts.

Conflict resolution is a common human activity, because conflict is inevitable in any human interaction. Conflict in a general sense is defined by the American Heritage Dictionary[6] as "A state of disharmony between incompatible or antithetical persons, ideas, or interests." A more traditional view of conflict is violence and the US Department of Defense[7] defines it as "An armed struggle or clash between organized groups within a nation or between nations in order to achieve limited political or military objectives." In the field of psychology conflict is "A struggle resulting from incompatible or opposing needs, drives, wishes, or demands;" this is from the Britannica Concise Encyclopedia[8]. As one can see conflicts appear on many different levels from the interpersonal level (the discipline of psychology) to global conflicts (the discipline of political science). Conflict resolution across all disciplines is understood as the peaceful and mutually satisfactory end or is significantly de-escalation of a conflict. Violence, deception or surrender is not considered conflict resolution as they only temporarily calm a conflict. According to Burton[9] conflict resolution has common characteristics across the various realms, therefore aspects of conflict resolution in one discipline can be applied to another. Burton states that three common traits that are, first detecting conflict, followed by reviewing goals and

finally determining the possible overcomes.

Conflict resolution is not explicitly addressed in most software development methodologies. Easterbrook[1] iterates that the avoidance of conflict is typically how conflicts are addressed. Conflicts are suppressed or ignored because there is no way to express them. Avoiding requirements conflicts may lead to poor design and may result in the failure of a project. The emerging field of requirements engineering (RE) attempts to address such conflicts. Easterbrook's[4] definition of RE is that it is the process where the purpose of a software system is determined based on the real-world needs of stakeholders affected by the system. Conflict resolution in requirements engineering consists of addressing the competing real-world needs of the people who have interest in the software project.

1.1 Problem Statement

There are several models, tools and methodologies in literature that support stakeholder collaboration as they resolve requirements conflicts and communicate these resolutions. Such need to be examined further in order to determine how well they support stakeholder collaboration in the early planning stage. Robinson[10] writes that stakeholder collaboration model needs to be able to detect conflict among requirements and be able to provide alternative solutions.

1.2 Purpose

The purpose of the research presented in this thesis is to review and analyze suitable models and tools that aid stakeholders collaboratively resolve requirements conflicts in the early planning stages. This is important because conflicts are rarely addressed in software development and the consequences of not dealing with them is great. Another rationale for our research is that the models that were studied were developed in a particular context and it should be determined whether the models will work in other contexts.

The models should have the capability to identify and visualize conflicts. They should also have the ability to propose viable solutions. Ideally, these models should be able to identify conflicts and generate potential solutions automatically, with minimal effort from a human operator. At the very least, the models should be an aid for people involved in the requirements negotiation process. Essentially, the models should help in **identifying, visualizing, and solving** conflicting requirements.

1.3 Methodology

To accomplish the purpose of the research, the research was split into two primary objectives, a basic research objective and an applied research objective. *The basic research objective was to study the modeling of conflict resolution in requirements*

engineering. This was done by performing a literature survey of several existing negotiation models that address requirements conflicts.

The applied research objective was to study how the existing models performed in contexts that are different from their initial one. Three of these models were studied extensively to satisfy this objective. These three models were chosen for further study because they were easier to apply and to comprehend.

We have adopted a case study approach in order to perform the investigation into the models. According to Kitchenham[11], a case study approach is taken to find the effects of technology in a typical situation, where the models are the technology under investigation. Bekker[12] states that case studies are especially appropriate when examining complex phenomenon such as conflict resolution in requirements engineering.

The case that was studied was a series of negotiation sessions between two stakeholders who had the task to determine the requirements of a new product. These negotiation sessions happened a few months before the application of the models, so the stakeholders had already resolved the conflicts – we shall call these resolved conflicts the model-free[1] results. In the postmortem[2] application of the models to the negotiation sessions, the answers to three primary questions were solicited. The

[1]Model-free refers to the original outcome of the case study, it's called the model-free because it is believed that no model was applied during the original negotiations

[2]Postmortem in this case refers to an analysis or review of a finished event

questions were:

1. Is the model useful in characterizing the conflicts?

2. Is the model able to visualize the conflicts?

3. Is the the model able to generate viable solutions?

The model-applied[3] results were analyzed and compared with the model-free results to gain more insights into the models and answer the research questions. The strengths and limitations of the models were documented in the analysis. Based on the strength and weaknesses of each model a hybrid model was developed.

1.4 Contribution

This research makes two key contributions to the discipline of software engineering. These contributions are:

1. A critical performance evaluation of existing conflict resolution models.

2. A hybrid model that leverages the strengths and addresses the limitations of the models examined in 1.

[3]Model-applied refers to results when a model is applied

1.5 Thesis Structure

This thesis is organized as follows. Chapter 2 reviews the literature that is relevant to our research project. This includes brief presentations on requirements engineering, conflict, and every model researched for this thesis. Chapter 3 describes the case study that was conducted to evaluate the applicability of the models and presents the results of the case study. Chapter 4 is the discussion chapter of the thesis, which includes the implications of the case study results, the observed limitations of the models and the proposal of a hybrid model. Chapter 5 draws conclusions and discusses the direction of future research.

Chapter 2

Literature Review

This chapter reviews all relevant background information for the research that was conducted. Section 2.1 discusses key concepts in the field of requirements engineering with a particular emphasis on multiple stakeholder development environment in the early planning stages of software development. Section 2.2 presents the main conflict-related concepts in software development. Section 2.3 details the conflict resolution tools and models that were researched.

2.1 Requirements Engineering

Brooks[3] stated that "the hardest single part of building a software system is deciding precisely what to build." The discipline of Requirements Engineering (RE) has emerged to address the conceptual work needed to determine what to build. A

formal definition offered by Easterbrook[4] is that RE is the "process of discovering the purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation." Basically the purpose of the system is found by seeking out the needs of "stakeholders." Some key concepts that capture the essence of RE are presented in the following section.

2.1.1 Main Concepts in Requirements Engineering

Stakeholder

The term stakeholder has been used in requirements engineering to describe all the participants related to a software system. As shown in the Britannica Concise Encyclopedia[8] this is a term borrowed from management and is defined as "a person or organization that has a legitimate interest in a project or entity." Boehm[13] offers a precise definition in RE where "stakeholders are individuals or organizations that stand to benefit from the success of the system or suffer from the failure of the system." There may be many stakeholders that are associated with a software system such as end-users, developers, customers and often the general public.

Requirements

Easterbrook[4] defines requirements as the description of a problem that should be solved. These descriptions must be elicited from the stakeholders and explicitly stated. Jackson[14] asserted that requirements do not directly describe the system, but are rather concerned with the the systems interaction with environment. Requirements describe changes in the environment one desires. For example for an aircraft control system, the environment is the airplane and requirements for this control system will be formulated in terms of the airplane's functions and not the underlying control system.

Faulk[5] said requirements are commonly categorized as functional and non-functional. Functional requirements deal with mappings between inputs into the system and a corresponding outputs. Non-functional requirements refer to all other constraints and may include performance, maintainability or safety. Brooks[15] has stated often that dealing with requirements is very difficult because software systems are extremely complex, making comprehension difficult.

2.1.2 Activities in Requirements Engineering

Requirements Engineering is a series of activities addressing requirements and includes "the elicitation, definition, modelling, analysis, specification and validation of what is needed from a system."[16] It is useful to divide the activities of RE into

two stages, which are the activities performed before delivering the requirement specification (RS) document and activities done after the RS. The pre-RS stage or the early planning stage is the most important stage of software development because the requirements of the system are determined. Brooks[3] has asserted that specifying requirements is crucial in software development because if the requirements are specified incorrectly the project can fail. The post-RS stage of RE involves managing requirements in the subsequent software development stages including design and coding. We are more concerned with pre-RS stage.

In the early planning stage requirements are elicited from the stakeholders. Because stakeholders have varying needs and desires requirements often come into conflict. Careful analysis of the requirements is needed to resolve such conflicts. Conflict resolution in requirements engineering is detailed in Section 2.2.

The goal of RE in the early planning stage of software development is to discover the purpose of a software system and communicate this in the RS document. Nuseibeh and Easterbrook[17] have identified five core activities in RE, which are: requirements elicitation, modeling and analysis, stakeholder agreement, communication and managing evolving requirements. Each activity is described in the following subsections.

2.1.3 Requirements Elicitation

The first activity is eliciting the requirements from the stakeholders of the future software system. The main difficulty in eliciting requirements is that the stakeholders often find it difficult to articulate their requirements. Brooks[3] has observed that stakeholders, especially clients, rarely know what they want, because software systems are very complex. Often it is useful for the stakeholders to think of a software system in terms of goals and tasks rather than detailed requirements. This leads to better comprehension of the purpose of the software system.

In order to properly extract ideas from stakeholders, requirements analysts have a variety of techniques at their disposal. The technique that is chosen depends on time and resources available and complexity of the problem. Some techniques include traditional data collecting, such as surveys and interviews, while others include prototyping and group elicitation sessions. After eliciting the raw requirements from the stakeholders they must be refined into a set of system requirements in subsequent RE activities.

2.1.4 Requirements Modeling and Analysis

The second stage in the RE process is the modeling and analysis of requirements. This involves analyzing the requirements statements elicited from the stakeholders and constructing a model of the product to be built. Models are abstract descriptions

that are amenable to interpretation[17]. Models are useful in representing software systems because they make comprehension of complex systems easier. There are several modeling techniques employed in RE, including goal-oriented models, techniques that model system behavior, and domain modeling, which describe the environment of the product.

2.1.5 Stakeholder Agreement

The third major activity is establishing agreement among stakeholders on the system requirements. This activity is crucial in our research because it involves conflict resolution. Every large engineering system built has multiple stakeholders. According to Nuseibeh[17] stakeholder requirements often come into conflict because stakeholders vary in goals and have a different perspectives of the environment. The goal of this activity is to obtain agreement among the stakeholders on a set of system requirements by resolving or reducing conflicts. Every stakeholder should be satisfied with the agreement.

In an ideal world all stakeholders should be fully satisfied with the system. However, in the real world there are both, in the terms used by Boehm[18], winners and losers. Building in a multi-stakeholder environment is akin to a zero-sum game, where the aggregate user satisfaction is always a constant. Some stakeholders would be less satisfied than others or, in some circumstances, adversely affected even though the

Table 2.1: Frequent Software Development Patterns

Proposed Solution	**"Winners"**	**"Losers"**
Quick, cheap, sloppy product	Developer, Customer	User
Lots of "bells and whistles"	Developer, User	Customer
Driving too hard a bargain	Customer, User	Developer

system was a success.

Table 2.1[18] is a simplified example of how "winners" and "losers" emerge. The three stakeholders in this example are the user who ultimately utilizes the product, the customer who pays for the software product and the developer who creates the software. While a sloppy and cheap product may benefit the customer, because of the low-cost, and the developer, since a software company may not have to put so much effort into sloppy product, it is of failure for the user because the product does not perform all the functions the user requires. A product with many "bells and whistles" may be nice for the user and the developer but not the customer. Users may benefit from advanced features, while a developer can charge higher fees for the extra features, which increases the cost for the customer. When customers and users are able to drive the price of a product down through bargaining the developer is losing out on revenue.

The notion of winners and losers is controversial. Boehm[19] states that if there

are any losers at all, the system is a failure. Indeed, Boehm believes everyone can be a winner when a good model that emphasizes collaborative processes is applied in the requirements analysis of a product. One goal of requirements engineering is to have all stakeholders be satisfied with and agree to the system requirements.

2.1.6 Requirements Communication

The fourth core activity of RE is the communication of requirements. The goal of the early planning stage is to effectively communicate the agreed system requirements. The requirements should be communicated to the stakeholders and also to the designers, implementers and testers of the software product. The vehicle for this communication is the Requirements Specification (RS) document. Kotonya and Somerville[20] state that a RS document contains a complete description of what the software will do, independent of implementation details. It is also an agreement among stakeholders about the requirement. There should be no requirements conflicts present in the RS. Most authors agree that the RS must have good quality.

Faulk[5] opines a good RS contains the following five properties:

1. Complete –The RS contains all information needed to begin implementation of the product.

2. Implementation Independent –The RS should have no design or implementation decision.

3. Unambiguous and Consistent –The RS cannot be open to interpretation by the stakeholders and cannot have contradictory statements.

4. Precise –The behaviour of the system must be defined exactly.

5. Verifiable –The document should be verifiable such that the RS sets benchmarks for testing.

The appendix contains excerpts from a RS document.

2.1.7 Evolving Requirements

The final activity is managing the evolution of requirements throughout the software development process. This activity is performed mostly in the post-RS phase of software engineering. Requirements often evolve even after the RS has been written. When a requirement or a group of requirements change, this change must be effectively communicated to the designers, implementers, testers or maintainers. According to Gotel[21], the most difficult part of managing evolving requirements is the requirements traceability problem, where documentation from the later development phases, such as comments in code, need to be linked to the requirements. Finding and maintaining these traceable links is the key to manage evolving requirements.

2.1.8 Problems in Requirements Engineering

Often times, software projects are delivered overbudget, behind schedule and with poor quality, or fail completely. These sort of failures are not isolated incidents but rather the norm. Many of the failures, delays, and budget overruns in software engineering can be traced directly to shortfalls in the requirements analysis process in the early planning stages from Easterbrook[4]. Faulk[5] asserts that the shortfalls can be directly attributed to the fact that requirements are difficult to address. Requirements are purely conceptual, therefore comprehension and communication is very difficult. It is very tempting to rush into implementation and dismiss addressing requirements altogether.

Failure to properly understand the system requirements in the early development stages leads to disasters in future stages of software development. It has been documented by Boehm[2] that the cost of fixing a requirement error in the early planning stage is much more cost-effective than in later stages. For example if it costs $1 to fix a requirement error in the early planning stages it will cost $10 in the coding stage and $100 in the maintenance stage. RE activities must be performed completely in the early planning stages to avoid project failure.

Another major problem that is often conflicting requirements remain unresolved[22]. Stakeholders' requirements often come into conflict because stakeholders have divergent goals and perspectives. These conflicts must be addressed and if possible solved

in the early planning stage. Section 2.2 provides information on conflict and conflict resolution in software development.

2.2 Conflict

Conflict, as it is understood today, is "a struggle resulting from incompatible or opposing needs, drives, wishes, or demands"[8] as defined by the Britannica Concise Encyclopedia. Conflict can be divided into two categories, internal conflict, which is a mental struggle within an individual, and interpersonal conflict, struggles among one or more individuals or groups. We are more concerned with interpersonal conflict, because requirements conflicts are of this nature. Interpersonal conflict appears in many forms and on many levels; from a dispute between two people, to countries at war over a resource, to conflicts within a organization such as a school or a multinational corporation.

When individuals and groups collaborate together to achieve a joint goal, such as a townhall meeting discussing traffic improvement or designing a new car model, interpersonal conflicts occur because the participants have divergent perspectives. The process of requirements engineering relies on collaboration among groups and individuals therefore interpersonal conflicts are prevalent.

Conflicts originate from many different sources, also many techniques have been developed to resolve conflicts. The following two subsections discuss the sources of

conflict and conflict resolution in both the general sense and specific to requirements engineering.

2.2.1 Sources of Conflict

The discipline of Conflict Theory addresses interpersonal conflict, and the current scholarship does not fully agree on the sources of conflict. Several writers in Conflict Theory have created different, but similar, models to describe the nature and sources of conflict. Deutsch[23] is an often cited theorist who determined five conditions in which conflicts arise. The five conditions were:

1. Control over Resources
2. Preferences and Nuisances, where one party's tastes impose on another
3. Differences of Values ("what should be")
4. Differences of Beliefs ("what is"), disputes over facts, information or reality
5. The nature of the relationship among the parties

Conflicts that occur in collaborative environments are called group conflicts. Robbins[24] has suggested that in organizational settings conflicts can arise from two primary sources: differing *goals* among the participants and varying participants' *perception* of the design domain. Even though several individuals have the same goal in mind,

conflicts may exist due to varying perceptions. Robbins[24] gives an example of a newly elected city manager, who promised improved garbage collection, started receiving complaints several months later that the service did not improve. Upon further investigation it was found that the conflict occurred because citizens perceived "improved service" to mean frequent collections, while the city manager regarded it as earlier, quieter and economical collections.

Conflicts that arise from varying goals are illustrated in the following example from Robbins[24]. The manager of an office required employees to fill out a longer form to document their weekly activities and the employees were against this change. The manager's goal was to gather more information and employees were concerned about not having enough time to fill out the form.

Conflict is an inevitable part of human collaboration including the activities of software development. Easterbrook[1] has identified some sources of conflicts that are applicable to the software development process. They include:

1. In large software projects the fact that the domain knowledge is spread over many developers causes more disagreements on how to solve the problem

2. Resource limitation, which includes limitations in money, personnel or hardware

3. Conflicting requirements due to the fact that there are multiple stakeholders with multiple and divergent needs and goals

4. Difference in perspective among the participants, one person may view the same concept or problem differently than another

On closer inspection, these four items are similar to the general sources of conflict, such as conflicts due to resource limitation and differences in values or goals.

2.2.2 Conflict Resolution

Why should requirements conflicts be resolved? Burton[9] has noted that the reasons for conflict resolution vary from discipline to discipline. For example, in political science hostilities between two nations should be ended so that the countries can begin trading with each other for their mutual benefit. In software engineering, requirements conflicts should be solved to avoid project failure later.

However, Burton[9] stated that not all conflicts need to be resolved. It is possible for certain conflicts that the best solution is not solving the conflict. When the cost, monetary or otherwise, of solving a conflict is much greater than not solving, then one may consider not solving the conflict.

A resolution method is required in order to solve most conflicts. Resolution methods borrow techniques from in various fields including mathematics, behavioural sciences and law. Strauss[25] has grouped resolution methods into three broad categories. The first category is *Collaborative* methods which include negotiations and education. The next category is *Competitive* methods where one party tries to achieve

maximum satisfaction without any regard for other parties. While this does not necessarily include hostility, the generally competitive methods produce winners and losers. The last category is *Third-Party Resolution*, usually occurs when the conflicting parties cannot find that resolution on their own. This may include appealing to an authority figure such as a judge or tossing a coin. According to Robinson[10] conflicts are resolved by using collaborative methods in the field of software and requirements engineering.

One challenge in conflict resolution is caused by the fact that in most software development methodologies conflicts are not addressed explicitly. Easterbrook[1] documents that the most common way software developers deal with conflicts is by not acknowledging them. Conflict suppression is a quick fix and may solve the problem in the short-term, but by avoiding the conflict it is probable that the conflict will arise in the later development stages and will be more expensive to solve. In small projects one may be able to get away with conflict suppression. Many times, conflict suppression leads to accepting one prevailing view at the expense of alternative views; therefore, only one dominant stakeholders view prevails.

In software development, especially requirements engineering, communication and collaboration among the various stakeholders is the best way to solve requirement conflicts. Robinson[10] has noted that conflict resolution in general consist of three basic processes:

1. Detection of conflict, which involves determining inconsistencies between requirements

2. Generation of various resolution options, where various conflict free requirements sets are presented

3. Choosing the correct resolution; the various options are evaluated and ranked, and the optimal solution is chosen

These three steps closely resemble the process Burton[9], a political scientist, has presented.

In summary, it is important to study conflict resolution techniques, because dealing with conflicts in the later stages of software development is more expensive than dealing with them early. As Brooks[3] has stressed, failure to address requirements conflicts can lead to project breakdown. Conflict resolution in software engineering is rapidly becoming an important field of study. Section 2.3 discusses various conflict resolution models.

2.3 Conflict Resolution Models

This section gives a description of each conflict resolution model that was researched. Three of the models are described in great detail and two other models are simply overviewed. The models reviewed extensively are the Utility Curves Model, the Win-

Win Model and the *i** Framework. The remaining models were Viewpoints and Negotiation Model for COTS Selection.

2.3.1 Utility Curves Model

The Utility Curves Model[26] was originally used in a requirements engineering setting by W.N. Robinson in the Oz project when he was at the University of Oregon in the early 1990s. The Oz project[10] is a support system for a collaborative requirements engineers who are designing a specification; it uses various quantitative and decision science methods to characterize and resolve conflicts among different stakeholders. Oz was extensively used to help design the University of Michigan General Library automated circulation system.

Eventhough the Oz system uses various different methods to detect and resolve conflicts, we chose to study solely the Utility Curves Model. This is because Oz as a whole has too many components and methods to understand and too cumbersome to apply. The Utility Curves Model is not difficult to understand, is simple to apply and because it is mathematical in nature it is not qualitative and subject to observer bias.

The basic premise of the Utility Curves Model is that for every issue each stakeholder possesses a unique utility curve function. That is to say stakeholders achieve a certain amount of satisfaction or usefulness, called simply utility, when a particular

requirement is attained. Each requirement is also assigned a relative weight. Because not all issues are of equal importance all parties can gain as more and more issues are added. It is possible to find a set of issues or requirements that satisfies all parties; this is called *integrative behavior*. If weights did not exist, it is called *distributed behavior*, the situation would liken to a zero-sum game, where one party's gain is another party's loss. This is because all issues are assumed to have equal importance. Both behaviours were studied in the thesis.

Each stakeholder possesses a unique utility curve for every requirement to be negotiated. Mathematically the utility curve can be represented by $U_a(x) = y$, where x is the attainment value for a particular requirement, a, and y is the utility value from 0 to 1.0. When the utility curves of a certain requirement is superimposed for two different stakeholders there may be a region of agreement in the mutual curves where it is possible that both stakeholders are satisfied.

In practice, utility curve functions are not used, because a requirement usually cannot be partially attained. The requirement attainment value is binary, it is either completely fulfilled or not fulfilled at all, therefore the model applies a utility value rather than a function to each requirement. The model also applies a weight to every stakeholder requirement for the study of integrated behavior. Every stakeholder or stakeholder team should assign utility and weight to each requirement. The model calculates a combination of requirements that will satisfy all stakeholders optimally

and appear in a specification document. Equation 2.1 is the formula used to calculate the utility of a particular combination when weights are considered (integrative behavior). Equation 2.2 is the formula used to determine the utility of a particular combination when the weights are not considered (distributed behavior). U^P refers to the utility of a combination, P. U_a and W_a referred to the utility and weight of requirement, a. There are m different requirements in one combination.

$$U^P = \frac{\sum_{a=0}^{m} W_a U_a}{\sum_{a=0}^{m} W_a} \tag{2.1}$$

$$U^P = \sum_{a=0}^{m} U_a \tag{2.2}$$

Once the utility value of every combination is calculated, the optimal combination must be chosen to be included in the specification of the system. There are two types of optimal combinations: one that produces the highest total utility and one that produces equal utility. For the highest utility criterion, "a good agreement should maximize the sum of the negotiating parties' utilities."[27] and the combination of the highest utility is noted. "The criterion of equality is fulfilled if the negotiated settlement gives equal utilities to all parties."[27]. The solutions based on both criteria should be examined when making a final decision.

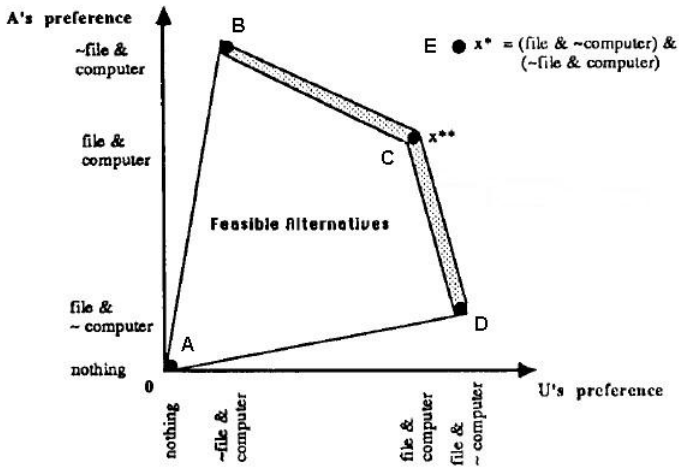

Figure 2.1: Searching for Ideal Alternatives employing Utility Curves

Example of Utility Curves Model

An example from Robinson[26] will be used to illustrate the Utility Curves Model. Consider the requirements negotiation of a university library filing system. There are two stakeholders a user (U) and a systems analyst (A). The user did not want to use a computer filing system and would have rather employed a card file system. The analyst, on the other hand, supported automated filing and was willing to develop a computer only filing system. There is a conflict between the stakeholders that needs to be solved.

The stakeholders' utility (preference in this case) is graphed in Figure 2.1. The utility of U is on the horizontal axis, while the utility of A is on the vertical axis. Each point is a possible solution, where point A is the trivial solution and neither

Weighted Requirements		Utility (0-1)		Weight			
R#	Requirement	U	A	(1-10)			
R1	Computer	0.40	1.00	10			
R2	~Computer	1.00	0.00	4			
R3	File	1.00	0.40	8			
R4	~File	0.20	0.80	2			
C#	Combination	U	A	Sum	Diff.		
A	R2+R4	0.37	0.13	0.50	0.23		
B	R1+R4	0.37	0.97	1.33	0.60		
C	R1+R3	0.67	0.73	1.40	0.07	highest	equal
D	R2+R3	0.67	0.18	0.84	0.49		

Figure 2.2: Weighted Requirements to Demonstrate Integrative Behaviour

nor file is employed. Point B is where the computer is employed and the file is not. Point C utilizes both computer and file, and point D employs the file but not the computer. Point E represents the ideal, but infeasible, solution where both stakeholders ideal requirements are met. Points ABCD form the region of feasible but undescribed alternatives, and the shaded area represent compromise alternatives. Point E lies outside the feasible region and therefore is an infeasible alternate. The "best" solution is point C, where both systems are employed.

This example will be quantified to demonstrate integrative and distributive negotiation behaviour. For both examples four separate requirements have been identified: computer, no computer, file, and no file. And from these individual requirements four feasible combinations have been identified: (A) no computer and no file; (B) computer and no file; (C) computer and file; and (D) file and no computer. In integrative

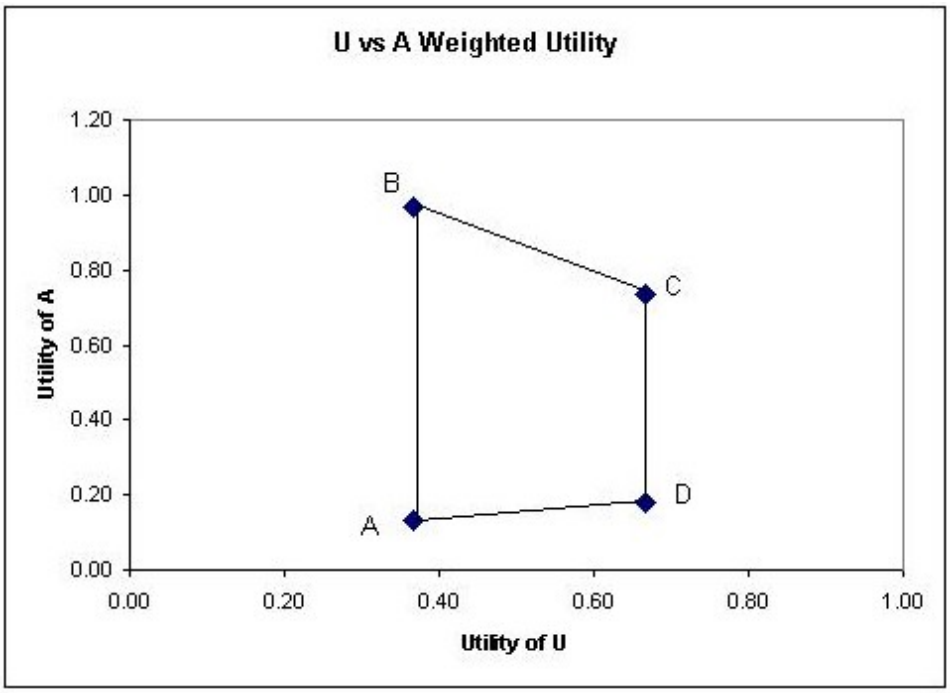

Figure 2.3: Feasibility Plot of Weighted Requirements to Demonstrate Integrative Behaviour

Unweighted Requirements		Utility (0-1)					
R#	Requirement	U	A				
R1	Computer	0.40	1.00				
R2	~Computer	1.00	0.00				
R3	File	1.00	0.40				
R4	~File	0.20	0.80				
C#	Combination	U	A	Sum	Diff.		
A	R2+R4	0.60	0.40	1.00	0.20		
B	R1+R4	0.30	0.90	1.20	0.60		
C	R1+R3	0.70	0.70	1.40	0.00	highest	equal
D	R2+R3	1.00	0.20	1.20	0.80		

Figure 2.4: Unweighted Requirements to Demonstrate Distributive Behaviour

behaviour every requirement has a weight as well as a utility associated with it. Figure 2.2 shows the utility and weight assigned to each requirement and the utility of each combination. The sum of utilities and the difference of utilities with calculated for each combination to determine the combination with the highest utility and most equal utility respectively. Combination C has both the highest and most equal utility. Figure 2.3 plots the feasible alternatives of the weighted requirements demonstrating integrative behaviour.

Figure 2.4 and Figure 2.5 demonstrate distributive behavior by removing weights from the individual requirements so that each requirement has equal importance. In this example only Combination C is again the optimal solution, which is the same combination when weights are considered. There are differences between weighted and unweighted requirements in the shape of the feasibility plots. One can see that

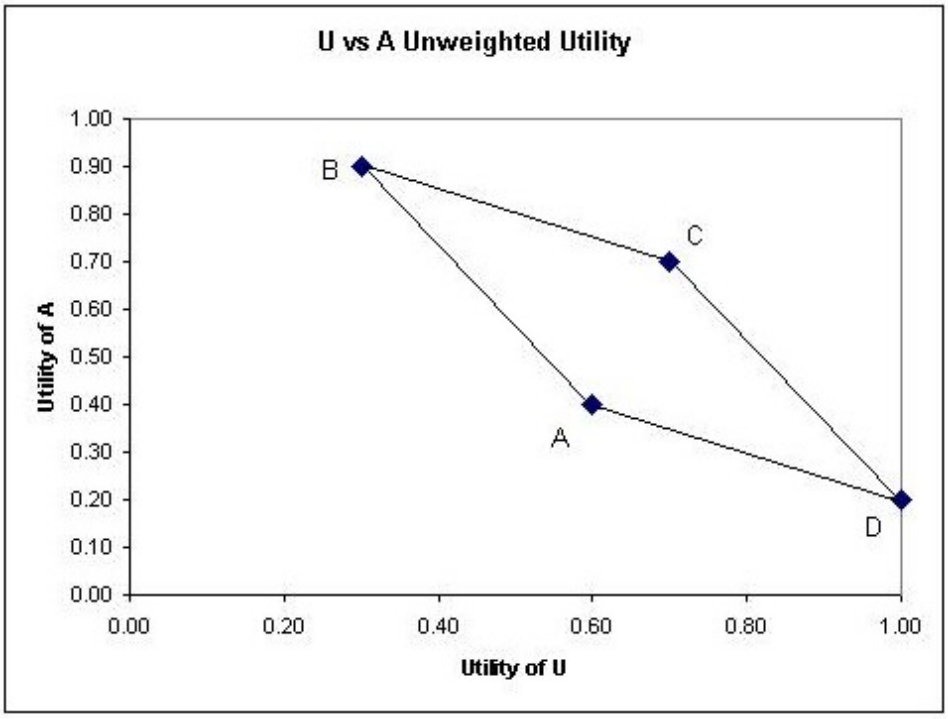

Figure 2.5: Feasibility Plot of Unweighted Requirements to Demonstrate Distributive Behaviour

in the unweighted feasibility plot, Figure 2.5, there are extreme distances between feasible points, while in the weighted feasibility plot, Figure 2.3, the distances are compact due to the use of weights. Extremes are expected in distributive negotiation behavior; it is akin to a zero-sum game where one party's gain is another party's loss by the same margin.

2.3.2 Win-Win Model

Win-Win model was developed in 1994 by Boehm as a combination of Theory W[19] and the Spiral Model[28] for software development, which was initially named Next Generation Process Model (NGPM)[18]. Theory W was developed by Boehm and Ross in 1989 both of the University of California, Los Angeles at the time. The Spiral Model was also developed by Boehm in 1988 as a software development tool. The Win-Win Spiral Model is presented as a model that can be applied to system design in a 1995 paper[29]. The Win-Win Model was applied to two major case studies in the late 1990s; one was University of Southern California digital library project, and the other was a satellite-ground-station project. In the digital library project graduate students acting as software developers worked with the library staff to resolve requirement conflicts that arise in the early planning stages. The ultimate goal is to digitize library archives.

The formal process is shown in Figure 2.6[18]. The first step calls for the identifi-

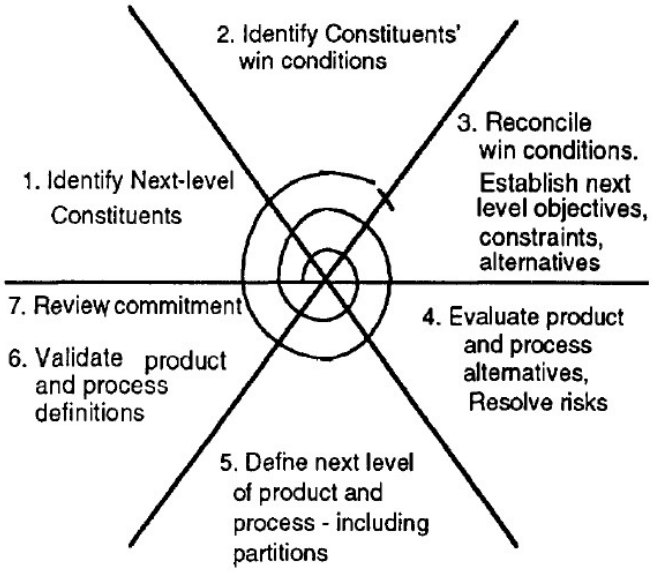

Figure 2.6: The Win-Win Spiral Model

cation of the next level constituents, which refers to stakeholders in the early planning stages. in subsequent iterations the next-level constituents may be the designers in the design stage or programmers in the implementation stage. In step two, the requirements that form the winning conditions for each stakeholder are identified. In the third step, the competing win conditions must be reconciled before moving on to the next level of software development. The algorithm used in step three reconcile win conditions is in the next paragraph. The first three steps form the basis of the Win-Win Model. Steps four through seven are validation stages before moving on to the next stage.

Figure 2.7 shows the general algorithm used in the Win-Win negotiation process.

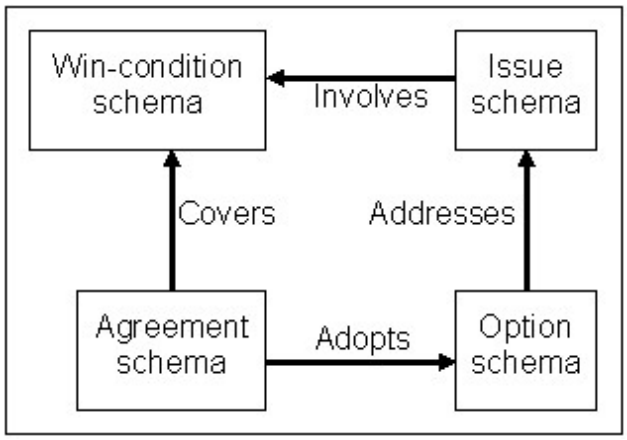

Figure 2.7: The Win-Win Negotiation Model

To begin with, each stakeholder creates his or her own Win Condition Schema, summarizing the stakeholders' requirements. The next step is to identify any conflicts that arise when comparing the various Win Condition Schemas. The conflicts are either identified by human operators or a knowledge-based tool. If there is a conflict, an Issue Schema is drawn up summarizing each conflict. For every issue an Option Schema is composed addressing the issue. The Option Schema are proposed by the stakeholders themselves. Stakeholders evaluate the options through negotiation or the use of a quantitative tool. An Agreement Schema is made by the stakeholders on the mutually satisfactory compromise that is made.

The Win-Win model itself does not identify and resolve conflicts; it is a general algorithm that guides stakeholders in their collaborative efforts. It requires other tools such as knowledge bases, analysis tools or human operators to identify conflicts,

and to evaluate alternative resolution option. We examined a number of such tools to be employed with the Win-Win Model. The tool used in Win-Win during the research was the Cost Benefit Analysis Method (CBAM)[30].

The following are other tools examined. Boehm introduced Quality Attribute Risk and Conflict Consultant (QARCC)[31] to be used as a compliment to Win-Win. QARCC relies on a knowledge base of a system that has been acquired over time. Using its knowledge base QARCC is able to identify conflicts, notify the correct stakeholders about the conflicts, and present viable solutions to the stakeholders in question. Other knowledge base tools used were COCOMO (COnstructive COst estimation MOdel)[2] and S-COST (Software Cost Option Strategy Tool)[32]. Both of these tools deal with cost requirement conflicts. One tool called, Multi-Criteria Preference Analysis[22], uses a formula very similar to the Utility Curves Model presented in Section 2.3.1 to rank different resolution options. Ultimately it was decided to employ CBAM as the tool to complement the Win-Win Model, because CBAM possessed a simple formula used to evaluate different resolution models. Multi-Criteria Preference Analysis also had a simple formula but because it was similar to the Utility Curves formulas it was rejected. QARCC, S-COST and COCOMO were also rejected because they both relied on knowledge base which the case study did not have.

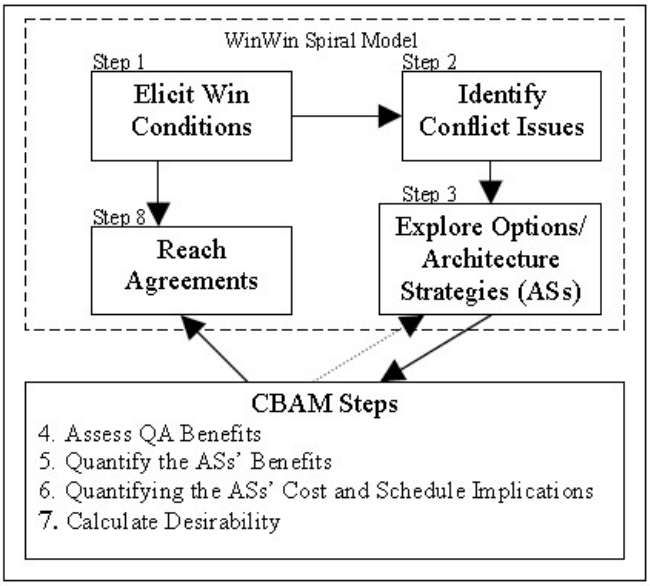

Figure 2.8: The Win-Win/CBAM Integrated Framework

Cost Benefit Analysis Method

The idea of combining the Cost Benefit Analysis Method (CBAM) and the win-win model was first proposed by In, Kazman and Olson in 2001[30]. CBAM itself was first introduced in 2001 by Kazman[33] as a method to evaluate different architectural strategies or requirements.[1] By integrating the Win-Win Model and CBAM, the strength of both methods can be exploited.

CBAM calls for identifying quality attributes, such as user-friendliness and good security. The requirements are evaluated based on their contribution to the quality

[1]The authors of CBAM use the term architectural strategy (AS), but in the thesis it is referred to as requirements.

attributes.

This new integrated framework is summarized in Figure 2.8[30]. The first three steps and step 8 are taken from the algorithm of the Win-Win spiral model. From Step 3, which is exploring different resolution options, to Step 8, which is reaching an agreement, CBAM is employed. In Step 4, the quality attributes are quantified by assigning them quality attribute scores (QAScore). Each quality attribute is given the value such that the sum of the values is 100. Quality attributes with more importance should be assigned greater value. For example:

```
Performance: 15

Security: 15

Modifiability: 30

Reliability: 25

Interoperability: 15
```

As one can see, Security is given a QAScore of 15. The quality attributes of a system may also be called system objectives or goals. In the fifth step, the benefit of a particular architectural strategy or requirement is calculated. The contribution the requirement makes towards each quality attribute must be quantified on a scale of 1 to +1. A +1 means that this requirement has a substantial positive effect on equality attribute (for example, requirement1 under consideration might have a substantial positive effect on quality attribute of Performance), a -1 means the opposite, and 0

is no contribution. The benefit of a particular resolution option, R_i, is calculated in Equation 2.3. The sum is over all the all the attributes, j.

$$Benefit(R_i) = \sum_j ((Contribution_{ij})(QAscore_j)) \tag{2.3}$$

The maximum benefit is upto +100, while the minimum benefit could be -100. For example, for resolution2 its contribution to quality attribute Performance is (-0.2), Modifiability (0.6), Interoperability (0.3) and no contribution to Security and Reliability.

```
Benefit(R2) = (-0.2)*15 + (0.6)*30 + (0.3)*15 = 20.5
```

In Step 6, the cost implication of each architectural strategy or requirement must be estimated. The literature does not specify any method to estimate cost, because it is assumed that each organization has its own cost estimation technique. These costs may not be in dollars and cents, but may be on a relative scale. Step 7 involves calculating the desirability of a particular requirement or architectural strategy, which is simply dividing the benefit by the cost. Equation 2.3 is the formula used to calculate the desirability of a particular option.

$$Desirability(R_i) = \frac{Benefit(R_i)}{Cost(R_i)} \tag{2.4}$$

For example, if the cost of implementing resolution2 was estimated to be 41, then

the desirability is $20.5/41 = 0.5$, which is a relative unit to be compared with other options.

2.3.3 *i** Framework

The ***i**** framework[34] was developed by Yu of the University of Toronto. The name ***i**** is formally called the *Notation of Distributed Intentionality*, which has many other applications within RE. The framework is a method to visualize the different actors in a software system and the relationships and dependencies among them. ***i**** creates graphs with actors as nodes and the dependencies between the actors as edges. This framework has both a graphical and formal representations. Once the requirements are characterized using ***i**** framework, the visualizations can be analyzed to detect conflicts and inconsistencies. The differing visualizations can be integrated using a variety of techniques. The ***i**** notation can be very useful in software systems that have many different actors, stakeholders or users.

There are two types of models within the ***i**** framework: *Strategic Dependency* (SD) and *Strategic Rationale* (SR). A good way to describe the difference between the two models is that in SD the dependencies shown in the graph are only external, while in SR they are both external and internal dependencies. In SD, the actors are shown as single atomic units with only their relationship with other actors described. In SR, the stakeholder interests and concerns are addressed and how the stakeholder

performs its internal tasks.

There are four types of dependent relationships that are based on resources being transferred, tasks being performed, goals to be attained and softgoals that should be achieved.

KHP Case Study

The ***i**** framework notation was first applied in a case study by Easterbrook in 2004[35]. The case study was a charity called, Kids Help Phone (KHP), who wanted to expand their services to the World Wide Web. The ***i**** notation was used to create graphical representations of users and their interdependencies. In the KHP case study, the main hypothesis that was being tested was that "modeling stakeholder viewpoints separately and then combining them leads to a richer understanding of the domain". Other hypotheses were to see if viewpoints can improve: backwards traceability, readability, capturing divergent and minority opinions, and the decomposition of large modeling tasks.

The case study was performed with two teams: team G for Global and team V for Viewpoints. Team G created a single view based on the 14 transcripts of interviews taken with the stakeholders of KHP. Team V created one model for each transcript and attempted to merge the different viewpoints into a single view. The views were created using the ***i**** framework. The paper unfortunately did not describe what

integration technique that team V used to merge the views.

The results of this case study were interesting. For the G team creating one single model out of the mountain of information they received proved to be quite difficult; they had to make slices of the model in order to make the graph readable. The V team on the other hand was unable to create a merged model, because of time constraints. But team V was able to demonstrate that backwards traceability from the model to the transcripts was improved using their method rather than the global method. Even though the teams were unable to create a single view, they have a better understanding of the strengths and weaknesses of decomposition of the tasks compared to a single monumental task.

2.3.4 Viewpoints

Viewpoints was developed in the early 1990s as a way to address multi-perspective multi-component system design[36]. It was initially designed for software engineering but is applicable to other engineering system. Viewpoints are used in the very early stages of software development. Viewpoints addresses the problems associated with designing systems with many components by many agents by partitioning the software development knowledge into manageable divisions. Partitioning is useful only if the relationships between different partitions are defined, so links are developed between the various viewpoints.

An individual viewpoint has an Owner associated and describes a particular system or subsystem. An Owner can be designer, systems analyst developer, customer or a user. Each viewpoint has the following slots: style, domain, specification, a work plan and a work record[20]. The style defines the representation that can be used to describe the subsystem, for example a state diagram or hierarchical tree are examples of representation style. The domain slot defines the scope of the viewpoint or the subsystem that is being covered. The specification is the actual description of the subsystem. The work plan slot is a list of actions that needs be taken for this viewpoint to work and finally the work record is a checklist for the work plan.

The main power of this model is that that differences between stakeholders' requirements can be reconciled in an iterative process where one repeatedly applies a set of basic rules to a pair of requirements until there are no inconsistencies between requirements. There could be many steps until one arrives at a place where there are no inconsistencies. A set of requirements can be represented by a state diagram which has a number of states a software system can be in and the transitions between the different states in a diagram[37].

There are several Viewpoints-Oriented models that can be used; some are used for design specification, while some are designed for requirement abstraction. Easterbrook[37] has stated that Viewpoints are a good elicitation tool for the identification of conflicts and believes that with experience Viewpoints can become a very useful conflict

resolution tool.

We decided not to apply any of the viewpoint models to a case study for a few reasons. Viewpoints requires a sufficient diversity of perspectives, while the context used in the case study has only two perspectives. The power of Viewpoints does not become used when there are only two perspectives. Also, Viewpoints requires deep knowledge of the application domain, which does not exist in the case study. The application knowledge was not developed in the case study because it was a new project.

2.3.5 Negotiation Model for Commercial Off-The-Shelf Selection

This model helps solve multi-agent requirements conflicts that arise when selecting commercial off-the-shelf (COTS) software products. The Negotiation Model for COTS Selection[38] (NeMo-CoSe) basically has the same iterative structure as the Win-Win model and employs the Analytical Hierarchy Process[39] (AHP), a Game Theory Model and a Knowledge Base (KB) in its process.

Like the Win-Win model NeMo-CoSe has an iterative process, the steps are repeated until the desired result is found. The preliminary step is to determine the evaluation criteria and which COTS product to evaluate. In the first step, stakeholders place relative weights on their evaluation criterion. In the second step, AHP is

applied to the evaluation criteria to rank the COTS. In the third step, the COTS rankings for each stakeholder are compared to identify conflicts. If conflicts arise we go to step four, otherwise the process is done. In step four, game theory model (coalition forming) is used to find an agreement option. Step five allows the stakeholders to reevaluate the agreement options or negotiate with other stakeholders and go back to the second step if needed. The authors of this model do not expect the steps to be followed rigidly.

It was decided that this model be rejected for further study and application to a case study. The case being studied dealt with customized software rather than COTS software, so NeMo-CoSe needed to be modified such that each individual software packages in COTS was seen as individual components in a single software package. Other reasons for rejection were that the case study had no developed knowledge base and that AHP was not appropriate when computing raw data.

2.4 Summary

In this chapter, a broad overview of Requirements Engineering activities was given. In the next section, the key concepts of conflict was discussed along with the sources of conflict and conflict resolution. The third section described in detail five different conflict resolution models, three of which were applied to a case study in Chapter 3.

Chapter 3

Case Study

Three of the models were selected for extensive research to gain more insights into the applicability of the model. We applied the models in a different context than in which each model was originally developed. The context was a set of requirements engineering activities namely a negotiation process between two stakeholder groups. The goal of the case study was to better understand how the models worked with respect to identifying, characterizing, visualizing and solving requirements conflicts. We also wanted to discover new capabilities and limitations of the models. The results of the case study will be used to construct a hybrid model that leverage the strengths and limitations of each individual model.

We decided to conduct the extensive research as a case study, because the field of requirements engineering is so complex that a case study must be performed as

stated by Bekker[12]. According to Kitchenham[11], a case study is appropriate when investigating the effects of technology in a typical situation, where the models are the technology under investigation and the context represents a typical situation.

Section 3.1 of this chapter gives an overview of the context chosen for the case study. Section 3.2 describes the data sets to which the models were applied. The three subsequent sections describe the application of the Utility Curves Model, Win-Win Model and the ***i**** Framework respectively. For each model, the exact procedure used to conduct the application is described followed by the full results of the application.

3.1 Overview

Six graduate student teams from three locations around the world participated in a collaborative software development exercise, between January and April 2005. The purpose of the exercise was to "experience development of software in geographically, distributed multi-cultural teams"[40]. More specifically, student teams acted as developers and clients to produce requirements specification documents using various communication media. This effort was organized by Dr. Daniela Damian[41] of the University of Victoria and Dr. Ban Al-Ani of the University of Technology, Sydney (Australia). There were in total six teams of four to five students; three teams were from the University of Victoria, two teams were from University in Australia and one team from the University of Bari in Italy. Each team acted as both a developer

for one negotiation process and a client for another, so there were six negotiation processes in total producing six different software products. The original purpose of these projects was to analyze and compare different communication media used during global requirements negotiations.

The reason these exercises in the case study were used was because the documentation in these negotiations sessions produced all the relevant data required for the case study and was readily available. Also requirements and conflicts were explicitly stated in the documentation. This is important because the more explicitly stated the requirements and conflicts are the less they are open to interpretation by the researcher.

For the purposes of this thesis, only one negotiation process between a Canadian team and an Australian team was studied. This negotiation process was chosen because we believed that this particular process had the greatest amount of documentation among the six projects. The software product that was being developed was a Media Distribution Platform (MDP), which is a system that is able to create and send bulk e-mail for the distribution of promotional material such as movie trailers. The e-mails are sent based on consumer information collected from various sources.

There were two stakeholders involved in the requirements negotiations, which were a Canadian Developer (Media Magic, MM) and an Australian Client (Vegemite Kids, VK). Figure 3.1 shows the different phases and deliverables of the requirements ne-

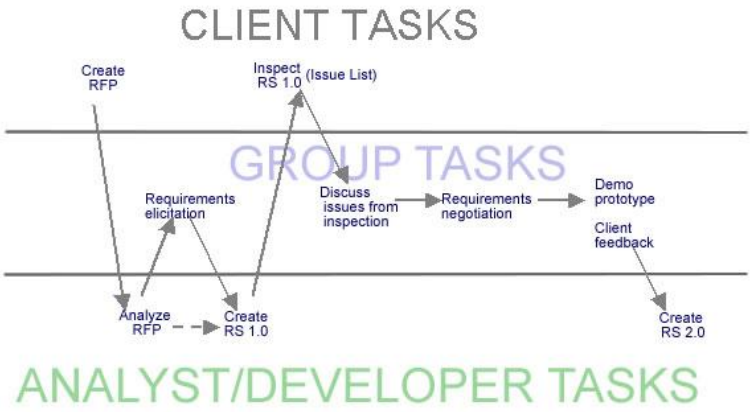

Figure 3.1: Phases of the Requirements Negotiation Process

gotiation process. Starting in the top left corner of the figure, the Client initially produced a Request for Proposal (RFP) for the Developer, who in turn analyzed the document. This was followed by a requirements elicitation session by videoconference in order for the Developer to gather more information to produce an initial requirements specification document (RS 1.0). In this session the first conflicts were becoming apparent. After this session, the Developer produced the RS 1.0 and sent it to the Client for inspection. The two teams then had an opportunity to "discuss issues from the inspection" of the RS 1.0 in an asynchronous online forum. Issues that were unresolved by the asynchronous session were discussed in a requirements negotiation session by videoconference. This was followed a week later by a prototype demonstration by the developer via videoconference. The prototype demonstration was an opportunity to discuss any outstanding issues before the developer produced

the final requirement specification document (RS 2.0). The RS 2.0 was the final deliverable from the Developer.

3.2 Data Sets

The data compiled for the entire process is stored in many forms. The recordings of the videoconferences where available in electronic form on DVDs and on camcorder tape. There was about six hours of videos. There was a mailing list that compiled all the e-mails sent between group members and across groups. There were formal documents as mentioned above, and the online asynchronous forum and there was a wiki-style website with more information. This data compilation[1] was used in order to perform the case study on this process.

The process of eliciting data from the videos involved watching and transcribing the videos for at least 60 hours over a period of four months. After transcription, the videos had to be referred to quite often to get an overall sense of the mood in the negotiation session. It should be noted that there was no interaction with any of the negotiators in order to stay unbiased.

The exact same data set was not used as input for each model. Different ranges of data had to be used for every model, although there was some overlap between the

[1]Data compilation or compilation will be the standard term for the data compiled for the negotiation process

three data ranges. Different input ranges were used primarily because the applicability ranges of each model is variant. In other words, each model accepts a different range of input data.

The Utility Curves Model accepts an initial requirements set as input from each stakeholder; the model assumes that all the requirements are in conflict with each other and evaluates the entire data set. The Win-Win model also receives an initial requirements set that comes with a qualifier stating which requirements are in conflict. Only the requirements in conflict are evaluated by the Win-Win model. Essentially, the Win-Win model uses a subset of the data used by the Utility Curves Model.

The initial requirements for each stakeholder group is displayed in Figure 3.2 for the Client and Figure 3.3 for the Developer. These initial requirements are used as input by the Utility Curves Model and the Win-Win Model.

The ***i**** Framework addressed a set of conflicts associated with assigning unique roles to different user types. These conflicts are based on a single requirement that calls for having different levels of authorization depending on the user. For example a manager has a higher authorization level then an employee, and also has more functions then an employee. The requirements these conflicts are based on are requirement 13 from Figure 3.2 and requirement 8 and 9 from Figure 3.3. Only a narrow subset of the initial requirements are used as the data input for the ***i**** Framework.

There is a small subset of the entire data range that can be applied to with all

Requirement #	Initial Requirements of the Client
1	Ability to send HTML format e-mails
2	Use external and internal sources for contact list
3	Manage Contact lists
4	Handling of Unsubscriptions
5	HTML editor
6	Ability to preview e-mails
7	Ability to report statistics
8	Controlled user access
9	Help section
10	Content is attached
11	Networked Virus Scanning Solution
12	Management of e-mail distribution
13	Manager only previews e-mails
14	Microsoft Office/Windows

Figure 3.2: Initial Requirements of the Client

Requirement #	Initial Requirements of the Developer
1	Bulk Import of consumer email lists
2	Manual editing of consumer data
3	Automated removal of invalid email addresses
4	Generation of consumer lists
5	Add/Edit/Remove content
6	Add/Edit/Remove templates
7	Preview
8	Assignment to Approval Managers
9	Authorization/Approval
10	Scheduling
11	Generate statistics report
12	Virus Scanning
13	Access Control
14	Help Section

Figure 3.3: Initial Requirements of the Developer

three models. The output data that is correlated with the input subset can be the basis of inter-model comparison.

3.3 Application of Utility Curves Model

The procedures this model used in the application are described first; followed by the complete results of this application.

3.3.1 Procedure

In the first step, the investigator[2] determined the explicit requirements of each stakeholder group based on the data compilation. These initial requirements were determined through the data compiled up to and including the requirements elicitation session. Obtaining data from the later stages of the process will skew the results, because up to the requirement elicitation session no bargaining is performed, only requirements are being discovered. At the end of this step there was a list of requirements from both stakeholder groups. Extracting these requirements was not easy because requirements are not necessarily explicitly stated. See section 4.1.4 about the difficulties in finding explicit requirements.

The next step was to divide the requirements into categories. The categories

[2]The investigator refers to the person who conducted the postmortem analysis and is also the author of the thesis

used closely follow the categories used by the stakeholder groups in the case study. Categories should be used to keep requirements that do not affect each other separate. At the end of the process there should be a set of requirements for each category. For example in Figure 3.4, the category is functional requirements and has 6 requirements (R1-R6), and in Figure 3.6, the category is consumer database requirements and has 4 requirements (R1-R4). It is also easier to interpret the results when the chosen requirements sets are separated into categories. Categories were not originally called for in Robinson's paper [26], but they were added for this case study to break up the problem into parts and to make it easier to understand the results.

The requirements in the case study are grouped into five categories, which were based on the components of the system. These categories are: Functional requirements, Consumer Database requirements, Manager Approval requirements, Performance Measures requirements, and User Interface requirements.

The next step was for the investigator to assign two utility values, U_a, one for each stakeholder group, and weight value, W_a, to each requirement in every category. The utility value is a number between 0 and 1.0; each requirement should have two utility values., one for each stakeholder group. There should be only one weight value for every requirement from 0 to 10. Ideally, these values should be inputted by the stakeholder groups or agents thereof, instead of the investigator. Unfortunately, in this case study the investigator assigns the values based on a set of guidelines set by

the investigator.

The following are a few tips on how values were assigned. The main notion is that the values are relative; utility and weight are not absolute values. If a requirement brings more value to one stakeholder group over another, then the utility value for one stakeholder group should be higher than the other stakeholder group's by that relative margin. If an issue had more importance or was more fundamental to the workings of the system, it should have a higher relative weight.

Once the requirements are placed into categories and assigned values, plausible combinations of requirements should be made. All possible combinations of requirements do not exist. There may be two or more requirements that are mutually exclusive, these requirements should not appear in the same combination. Some requirements need to be included in every plausible combination; then there cannot be a combination that omits a certain requirement. For example in Figure 3.4, there are six requirements, R1 to R6, and four plausible combinations, C1 to C4. C1 consists of R1, R2, R3 and R5. R1 and R2 are so important they had to be included in every combination. Without R1 and R2 the software system cannot run. Also, R3 and R4 are mutually exclusive, they don't appear in the same combination anywhere, and so are R5 and R6. Making plausible combinations of requirements may not be a necessary step, if there are neither mutually exclusive nor prerequisite requirements. In that case, the computer can evaluate all possible combinations.

Equations 3.1 and 3.2 were presented earlier in Section 2.3.1 as Equations 2.1 and 2.2 respectively. Using Equation 3.1 the investigator calculated the weighted utility value, U^P, for every plausible combination, P, and Equation 3.2 was used to calculate the unweighted utility value, U^P, for each combination, P. Weight, W_a, and utility, U_a, of each requirement, a, in a combination, P, were inputted into Equation 3.1, while only the utility, U_a, of each requirement, a, in a combination, P, were inputted into Equation 3.2. Every combination has m requirements. The utility of each combination must be calculated for each stakeholder group – the Client and the Developer.

$$U^P(x) = \frac{\sum_{a=0}^{m} W_a U_a(x)}{\sum_{a=0}^{m} W_a} \tag{3.1}$$

$$U^P(x) = \sum_{a=0}^{m} U_a(x) \tag{3.2}$$

The data was further refined by calculating the sum of utilities and the difference of utilities for every combination. This is to determine which plausible combination satisfies both stakeholder groups equally and those that have the highest aggregate utility. To determine the combination that equally satisfies both stakeholder groups, the investigator finds the combination that has the lowest difference in utility between the stakeholder groups. If there is a tie, then the combination with the highest sum of utilities is the best combination based on the equality criteria. The combination

with the highest aggregate utility can be found by determining the highest sum of utilities. Again if there is a tie, then the combination with the lowest difference of utility is chosen. Each category can have up to two combinations based on the different criteria.

With these results, stakeholder groups and their negotiators use the optimum solutions indicated by the model. They can either accept or discuss these solutions.

3.3.2 Results

Figure 3.4 through Figure 3.13 show the results of the application of the Utility Curves Model. There are two figures for each requirement category, one for weighted results and another for unweighted results. Every figure has up to two solutions, one based on the highest criteria and the others based on the equality criteria. Sometimes there is one combination that fits both criteria. In each figure, there are a list of requirements in the top half and a list of combinations in the bottom half. Beside each requirement there is a brief description of the requirement, two utility values, one for the Client and one for the Developer, and a weight if it is Weighted Utility Curves. For each combination there are the contents of the combination (For combination C1 in Figure 3.4 1+2+3+5 refers to requirements R1, R2, R3 and R5 are included in C1), two utility values, the sum of the two utility values, and the difference between the utility values. The combinations with the highest utility (greatest sum), equal

Weighted Functional Requirements		Utility (0-1)		Weight				
R#	Requirement	Client	Developer	(1-10)				
R1	Ability to send HTML format e-mails	1.00	1.00	10				
R2	Virus Scanning of e-mails	1.00	1.00	10				
R3	Content is attached	0.50	0.80	8				
R4	Content is URLs	1.00	0.80	8				
R5	External template editor	0.50	1.00	5				
R6	Internal template editor	1.00	0.10	5				
C#	Combination	Client	Developer	Sum	Diff.			
C1	1+2+3+5	0.80	0.95	1.75	0.15			
C2	**1+2+4+5**	**0.92**	**0.95**	**1.88**	**0.03**	**highest**	**equal**	**model-free**
C3	1+2+3+6	0.88	0.82	1.69	0.06			
C4	1+2+4+6	1.00	0.82	1.82	0.18			

Figure 3.4: Weighted Functional Requirements for the Application of the Utility Curves Model

utility (least difference) and combination determined by the model-free negotiators are noted.

Figure 3.4 and Figure 3.5 are the results of the Utility Curves Model for the functional requirements category. The requirements in this category are associated with the features of the e-mails that are being distributed.

In Figure 3.4, we assigned a utility of 1.00 for both client and developer for both R1, the ability to send HTML formatted e-mail, and R2, virus scanning of e-mails, because both stakeholders wanted these requirements in their initial requirements set. These two requirements were assigned the highest weight of 10 stressed the importance of these two requirements in the first video conference session.

For R3, when the idea of attaching content to the e-mails was discussed, the client

Unweighted Functional Requirements		Utility (0-1)						
R#	Requirement	Client	Developer					
R1	Ability to send HTML format e-mails	1.00	1.00					
R2	Virus Scanning of e-mails	1.00	1.00					
R3	Content is attached	0.50	0.80					
R4	Content is URLs	1.00	0.80					
R5	External template editor	0.50	1.00					
R6	Internal template editor	1.00	0.10					
C#	Combination	Client	Developer	Sum	Diff.			
C1	1+2+3+5	0.75	0.95	1.70	0.20			
C2	**1+2+4+5**	**0.88**	**0.95**	**1.83**	**0.08**	**highest**	**equal**	**model-free**
C3	1+2+3+6	0.88	0.73	1.60	0.15			
C4	1+2+4+6	1.00	0.73	1.73	0.28			

Figure 3.5: Unweighted Functional Requirements for the Application of the Utility Curves Model

seemed less keen on the idea then the developer and the developer seemed in different. Therefore we assigned 0.5 for the client and slightly more 0.8 for the developer. The client seemed much more enthusiastic about providing URL links to the content, R4, then attaching content and were assigned 1.0, well the developer, still indifferent to how content is delivered was assigned 0.8. We believed the method of delivery of content was slightly less important than sending HTML e-mails therefore we assigned a weight off 8 to R3 and R4.

It seemed that the importance of the template editor is less than method of content delivery, therefore assigned the two template editor requirements, R5 and R6, weights of 5. The main issue was whether to create templates in an external editor and upload them into the system or have a built-in editor. The developer strongly felt that the

external template editor has the best solution and were assigned 1.0 and the client were disappointed with the solution and resigned 0.5. For R6, the client wanted the internal template editor so were assigned 1.0, while the developer refused to accept this solution so were assigned 0.1.

The result in Figure 3.4 is that the combination producing the highest and equal utility is C2.

For Figure 3.5 the same rationale as Figure 3.4 was used to assign utility. The result in Figure 3.5 is that the combination producing the highest and most equal utility is C2.

C2 is the combination of requirements chosen by the model-free negotiators in both Figure 3.4 and Figure 3.5.

Figure 3.6 and Figure 3.7 are the results of the Utility Curves Model for the consumer database requirements category. These requirements are related to the maintenance of the contact list database.

The first requirement, R1, the uploading, editing and removal of contacts were present in both stakeholders initial requirement set and therefore 1.0 was assigned for both. Since the management of contacts is a major system requirement it was assigned a weight of 10.

As for generating contact information from reporting metrics, the client was assigned 1.0 and because the developer did not put too much opposition to this require-

Weighted Consumer Database Requirements		Utility (0-1)		Weight			
R#	Requirement	Client	Developer	(1-10)			
R1	Uploading, Editing, and Removing contacts	1.00	1.00	10			
R2	Generating contact list from metrics	1.00	0.85	5			
R3	Fixed address format	0.25	1.00	8			
R4	Delimited text address format	0.80	0.70	8			
C#	Combination	Client	Developer	Sum	Diff.		
C1	1+2+3	0.74	0.97	1.71	0.23		
C2	**1+2+4**	**0.93**	**0.86**	**1.79**	**0.07**	**highest**	**model-free**
C3	1+3	0.67	1.00	1.67	0.33		
C4	**1+4**	**0.91**	**0.87**	**1.78**	**0.04**	**equal**	

Figure 3.6: Weighted Consumer Database Requirements for the Application of the Utility Curves Model

Unweighted Consumer Database Requirements		Utility (0-1)					
R#	Requirement	Client	Developer				
R1	Uploading, Editing, and Removing contacts	1.00	1.00				
R2	Generating contact list from metrics	1.00	0.85				
R3	Fixed address format	0.25	1.00				
R4	Delimited text address format	0.80	0.70				
C#	Combination	Client	Developer	Sum	Diff.		
C1	1+2+3	0.75	0.95	1.70	0.20		
C2	**1+2+4**	**0.93**	**0.85**	**1.78**	**0.08**	**highest**	**model-free**
C3	1+3	0.63	1.00	1.63	0.38		
C4	**1+4**	**0.90**	**0.85**	**1.75**	**0.05**	**equal**	

Figure 3.7: Unweighted Consumer Database Requirements for the Application of the Utility Curves Model

ment they are assigned 0.85. Because this was a minor requirement we assigned it the weight of 5.

The address format was a contentious issue in one of the video conferences therefore 8 was the weight of both R3 and R4. The developer favored having a fixed address format therefore they were assigned 1.0, while the client was against fixed address format because contact information comes from many different sources therefore their utility was 0.25. Delimited text was somewhat favored by both parties with the client favoring it slightly more than the developer. The utility of the client and the developer was assigned to 0.8 and 0.7 respectively.

The combination that produced the highest utility was C2, while the combination with equal utility was C4.

The same rationale is used to assign utility values in Figure 3.7 as in Figure 3.6. In Figure 3.7 the combination producing the highest utility was C2, while the combination with equal utility was C4.

The combination chosen by the negotiators who were model-free was C2 for the consumer database category.

Figure 3.8 through Figure 3.9 are the results of the Utility Curves Model for the approval requirements category. These requirements deal with the pre-approval of e-mails before they are sent.

Both the stakeholders wanted in ability to preview e-mails before they are sent so

Weighted Approval Requirements		Utility (0-1)		Weight			
R#	Requirement	Client	Developer	(1-10)			
R1	Ability to preview e-mails	1.00	1.00	10			
R2	Manager only previews e-mails	1.00	0.80	8			
R3	Manager previews and edits e-mail	0.75	1.00	8			
R4	Employee selects approval manager	0.50	1.00	3			
R5	System selects approval manager	1.00	0.85	3			
C#	Combination	Client	Developer	Sum	Diff.		
C1	**1+2+4**	**0.93**	**0.92**	**1.85**	**0.00**	**equal**	
C2	1+3+4	0.83	1.00	1.83	0.17		
C3	**1+2+5**	**1.00**	**0.90**	**1.90**	**0.10**	**highest**	**model-free**
C4	1+3+5	0.90	0.98	1.88	0.07		

Figure 3.8: Weighted Approval Requirements for the Application of the Utility Curves Model

Unweighted Approval Requirements		Utility (0-1)					
R#	Requirement	Client	Developer				
R1	Ability to preview e-mails	1.00	1.00				
R2	Manager only previews e-mails	1.00	0.80				
R3	Manager previews and edits e-mail	0.75	1.00				
R4	Employee selects approval manager	0.50	1.00				
R5	System selects approval manager	1.00	0.85				
C#	Combination	Client	Developer	Sum	Diff.		
C1	1+2+4	0.83	0.93	1.77	0.10		
C2	1+3+4	0.75	1.00	1.75	0.25		
C3	**1+2+5**	**1.00**	**0.88**	**1.88**	**0.12**	**highest**	**model-free**
C4	**1+3+5**	**0.92**	**0.95**	**1.87**	**0.03**	**equal**	

Figure 3.9: Unweighted Approval Requirements for the Application of the Utility Curves Model

they were both assigned utility of 1.0. Because this is a major system requirements this requirement was assigned the highest weight possible of 10.

R2 and R3, was a contentious issue and were both assigned a weight of 8. The client believed that the manager should only preview e-mails and not edit them, while the developer favored managers being able edit e-mails as well as preview them for ease of development. For R2 the client had a slightly larger utility than the developer, while for R3 the developer has a larger utility then the client.

R4 and R5 were a minor issue and was assigned the weight of 3. The developer favored the employee selecting the approval manager, utility 1.0, while the client did not, but was not completely opposed to the idea that's why the client was assigned utility of 0.5. The client really wanted automatic selection of approval manager so they have a utility of 1.0 for this requirement. The developer did not mind having this requirement so they were assigned a utility of 0.85.

The combination that produced the highest utility was C3, while the combination with equal utility was C1.

The same rationale is used to assign utility values in Figure 3.9 as in Figure 3.8. In Figure 3.9 the combination producing the highest utility was C3, while the combination with equal utility was C4.

The combination chosen by the negotiators, who were model-free, was C3 for the approval category.

	Weighted User Interface Requirements	Utility (0-1)		Weight		
R#	Requirement	Client	Developer	(1-10)		
R1	Use web browser to view templates	1.00	0.75	10		
R2	Use other desktop application to view templates	0.40	0.75	10		
R3	Wiki help section	0.70	0.60	8		
R4	Templates created in HTML	0.90	0.80	5		
R5	Templates created in XML	0.50	0.80	5		
C#	Combination	Client	Developer	Sum	Diff.	
C1	**1+3+4**	**0.87**	**0.71**	**1.58**	**0.17**	**model-free**
C2	2+3+4	0.61	0.71	1.32	0.10	
C3	1+3+5	0.53	0.71	1.23	0.18	
C4	2+3+5	0.53	0.71	1.23	0.18	
C5	**1+4**	**0.97**	**0.77**	**1.73**	**0.20**	**highest**
C6	2+4	0.57	0.77	1.33	0.20	
C7	**1+5**	**0.83**	**0.77**	**1.60**	**0.07**	**equal**
C8	2+5	0.43	0.77	1.20	0.33	

Figure 3.10: Weighted User Interface Requirements for the Application of the Utility Curves Model

Figure 3.10 through Figure 3.11 are the results of the Utility Curves Model for the user interface requirements category.

The first major requirement choice in this category the stakeholders had to make was whether to view templates in a Web browser or on other desktop application. Since this was a major issue both requirements were assigned a weight of 10. The client wanted the Web browser therefore may have a utility of 1.0 while the developer did not care whether R1 or R2 was chosen so was assigned 0.75. The client did not favor desktop application and was assigned 0.4 utility and again the developer was assigned 0.75 due to the developer's indifference.

Both client and developer liked the idea of the wiki help section, but neither was

Unweighted User Interface Requirements		Utility (0-1)				
R#	Requirement	Client	Developer			
R1	Use web browser to view templates	1.00	0.75			
R2	Use other desktop application to view templates	0.40	0.75			
R3	Wiki help section	0.70	0.60			
R4	Templates created in HTML	0.90	0.80			
R5	Templates created in XML	0.50	0.80			
C#	Combination	Client	Developer	Sum	Diff.	
C1	**1+3+4**	**0.87**	**0.72**	**1.58**	**0.15**	**model-free**
C2	2+3+4	0.67	0.72	1.38	0.05	
C3	1+3+5	0.53	0.72	1.25	0.18	
C4	2+3+5	0.53	0.72	1.25	0.18	
C5	**1+4**	**0.95**	**0.78**	**1.73**	**0.18**	**highest**
C6	2+4	0.65	0.78	1.43	0.13	
C7	**1+5**	**0.75**	**0.78**	**1.53**	**0.03**	**equal**
C8	2+5	0.45	0.78	1.23	0.33	

Figure 3.11: Unweighted User Interface Requirements for the Application of the Utility Curves Model

enthusiastic about it therefore the client was assigned 0.7 and the developer who we seemed even less keen was assigned 0.6. The help section requirement was given a weight because although important was not as important as the issue of template viewer.

A minor issue of coding language arose therefore they were assigned a weight of five. The developer did not mind whether XML or HTML was used therefore they were assigned a utility of 0.8 for both requirements. The client seems to favor HTML or XML therefore they are assigned 0.9 for the HTML requirement while being assigned 0.5 for the XML requirement.

The combination that produced the highest utility was C5, while the combination

with equal utility was C7.

The same rationale is used to assign utility values in Figure 3.11 as in Figure 3.10. In Figure 3.11 the combination that produced the highest utility was C5, while the combination with equal utility was C7.

The combination chosen by the model-free negotiators was C1 for the user interface category.

Figure 3.12 and Figure 3.13 are the results of the Utility Curves Model for the performance reporting requirements category. For feedback and accountability purposes both stakeholders required some sort of performance reporting mechanism. This category contains requirements associated with this mechanism.

The first requirement, R1, was the ability to report 4 basic metrics, which are total emails opened, the number of unique emails opened, total bounced emails and the number of hyperlink hits. This requirement was given the weight of 10 because of the importance of these metrics for feedback. The client had a utility of 1.0 because they proposed these metrics. The developer was given a utility of 0.8 and because they were slightly hesitant in accepting these metrics.

The reporting of subscriptions was a minor requirement therefore was assigned a weight of 5. Since the client proposed this requirement, but as an afterthought they were given the utility of 0.9. The developer was not happy when they first heard of the requirement therefore they had a utility of 0.5.

Weighted Performance Reporting Requirements		Utility (0-1)		Weight			
R#	Requirement	Client	Developer	(1-10)			
R1	4 basic metrics	1.00	0.80	10			
R2	Reporting of Unsubscriptions	0.90	0.50	5			
R3	scheduled Accessing of reports	0.50	1.00	8			
R4	scheduled & ad hoc Accessing of reports	1.00	0.75	8			
R5	Fixed Report Structure	0.25	0.75	5			
R6	Customizable report structure	0.90	0.25	5			
R7	Tracking of accountability	0.75	0.50	3			
R8	Reporting on success or failure of distributions	0.75	0.50	3			
C#	Combination	Client	Developer	Sum	Diff.		
C1	1+2+3+5	0.71	0.79	1.50	0.09		
C2	1+2+4+5	0.85	0.63	1.48	0.21		
C3	1+2+3+6	0.82	0.71	1.53	0.12		
C4	1+2+4+6	0.96	0.63	1.60	0.33		
C5	1+3+5	0.66	0.86	1.52	0.20		
C6	1+4+5	0.84	0.77	1.61	0.07		
C7	**1+3+6**	**0.80**	**0.75**	**1.55**	**0.05**	**equal**	
C8	**1+4+6**	**0.98**	**0.66**	**1.64**	**0.32**	**highest**	**model-free**
C9	1+2+4+6+7+8	0.93	0.61	1.54	0.32		

Figure 3.12: Weighted Performance Reporting Requirements for the Application of the Utility Curves Model

Unweighted Performance Reporting Requirements		Utility (0-1)					
R#	Requirement	Client	Developer				
R1	4 basic metrics	1.00	0.80				
R2	Reporting of Unsubscriptions	0.90	0.50				
R3	scheduled Accessing of reports	0.50	1.00				
R4	scheduled & ad hoc Accessing of reports	1.00	0.75				
R5	Fixed Report Structure	0.25	0.75				
R6	Customizable report structure	0.90	0.25				
R7	Tracking of accountability	0.75	0.50				
R8	Reporting on success or failure of distributions	0.75	0.50				
C#	Combination	Client	Developer	Sum	Diff.		
C1	1+2+3+5	0.66	0.76	1.43	0.10		
C2	1+2+4+5	0.79	0.58	1.36	0.21		
C3	1+2+3+6	0.83	0.64	1.46	0.19		
C4	1+2+4+6	0.95	0.58	1.53	0.38		
C5	1+3+5	0.58	0.85	1.43	0.27		
C6	**1+4+5**	**0.75**	**0.77**	**1.52**	**0.02**	**equal**	
C7	1+3+6	0.80	0.68	1.48	0.12		
C8	**1+4+6**	**0.97**	**0.60**	**1.57**	**0.37**	**highest**	**model-free**
C9	1+2+4+6+7+8	0.88	0.55	1.43	0.33		

Figure 3.13: Unweighted Performance Reporting Requirements for the Application of the Utility Curves Model

R3 and R4 deal with the accessing of reports. This seemed to be a major issue with the stakeholders therefore was weighted 8. The client wanted to access of the reports on a scheduled basis and at any time (ad hoc), while the developer was not so keen on ad hoc accessing. The developer preferred only scheduled accessing while the client did not accept this requirement.

R5 and R6 had the weight of 5 because it is not as important as R1. Both these requirements deal with the structure of the performance reports. The utilities were assigned such that it reflects the fact that the client preferred customizable report structure while the developer preferred the fixed report structure. It

The last two requirements were minor and are given weight of 3. Both requirements were proposed by the client in one of the negotiation sessions therefore the client is assigned a utility of 0.75, while the developer is assigned a lower utility of 0.5.

The combination that produced the highest utility was C8, while the combination with equal utility was C7.

The same rationale is used to assign utility values in Figure 3.13 as in Figure 3.12. In Figure 3.13 the combination that produced the highest utility was C8, while the combination with equal utility was C6.

The combination chosen by the negotiators, model-free, was C8 for the performance measures category.

3.4 Application of Win-Win Model

In the case study, the key item to be determined is whether applying the Win-Win model to the negotiation sessions would alter the model-free results or whether the model-free results will remain unchanged. Solutions are found using the Cost Benefit Analysis Method (CBAM)[33]. The procedures of the Win-Win model application is discussed first and the full results of the application is detailed next.

3.4.1 Procedure

The first step is to identify the conflicts in the requirements. In this particular case study, the investigator used the logs of the asynchronous forum to identify the conflicts, because the conflicts were explicitly stated in the logs. Each stakeholder team worked individually to compile a list of issues. The stakeholder groups listed many perceived conflicts that were not actual requirements conflicts. There were 112 issues compiled by the stakeholder groups, and the investigator had to determine which of these issues were actual requirements conflicts. The issues that were excluded were: issues that should be dealt in the design stage and not the early planning stage, objects missing from diagrams in the first Requirement Specification, definition clarification issues, repeat conflicts, misunderstood meaning, and there were issues that were mistaken as conflicts where none existed. The conflict that were chosen to be studied were requirements free of implementation detail.

In each stage of the Win-Win method, the investigator determined a list of viable solutions (usually two or three solutions) per conflict identified. These possible resolutions were found in the logs of the asynchronous forum and the requirements negotiation session. The best solution should be found using CBAM. The investigator found the alternative solution options in the discussion portion of the asynchronous online forum. Each solution should be assigned a cost value between 10 and 100, and a quality assurance value from -1 to +1 for every objective. The cost value is a relative value of the cost of implementing the solution, where 10 is assigned when there is no cost and 100 is the highest cost. In the early stages of the case study, the client identified four objectives for the project. They were:

A Ability to distribute emails with video streams to contact lists

B Simple interface designed for effective use

C Ability to send emails for advertising purposes

D Accountability of activities performed in order to prevent misuse

Each stakeholder group needed to have QAScores assigned to each objective based on importance. The QAScores are a sign that such that the sum of the QAScore at up to 100. The QAScore are assigned at the beginning of the process and remains constant. Each conflict has between two and four alternative options that must be evaluated; these alternatives solutions were proposed by the negotiators. According to

CBAM every solution should be rated on a scale of -1 to +1 on how it will contribute to each objective, Cont_{ij}. A rating of +1 means the solution greatly contributes to the objective, -1 means the solution negatively contributes to the objective, and 0 means the solution has no effect on that particular objective of the system. once the cost and the quality assurance values are determined one needs to calculate the mean desirability of each alternative solution, in order to determine the best solution. Equation 2.3 and Equation 2.4 are the two formulas employed by CBAM. these equations are presented again as Equation 3.3 and Equation 3.4.

$$Benefit(R_i) = \sum_j (Cont_{ij} x QAscore_j) \tag{3.3}$$

$$Desirability(R_i) = \frac{Benefit(R_i)}{Cost(R_i)} \tag{3.4}$$

The option with the highest mean desirability should be noted, and it should also be noted whether it is clearly more desirable or marginally more desirable than the other options. A clear winner is the option whose mean desirability is greater than the mean desirability of the second-highest option by a difference of one.

3.4.2 Results

The results of the application of the Win-Win Model are shown from Figure 3.14 through Figure 3.17. Each figure lists all the conflicts of a particular iteration (the

first iteration is split into two figures). At the top of every figure the objectives of the system and the QAScores of each objective from the point of view of each stakeholder. For example in Figure 3.14, the client QAScore for objective A was 30, objective B 30, objective C 25, and objective D 15. For the developer on the other hand the QAScore for objective A was 50, objective B 15, objective C 15, and objective D 20.

For each conflict listed in the figures, there is a description of the conflict, followed by a menu of viable resolution options and the numbers inputted into the CBAM equations and numerical output for each solution option are displayed. The optimal solution calculated by CBAM is indicated as "winner". If the mean desirability of the optimal solution is greater than one over the mean desirability of the second-highest option, the solution is indicated as a "clear winner", otherwise it is called a "marginal winner." For each conflict the resolution option negotiated by the model-free negotiators is indicated by "m-f" if a resolution is found. For example in Figure 3.14 the first conflict has the label 1.1 and is described as "product has no interface to create templates. This conflict refers to requirement 3 for the Client from Figure 3.2 while for the Developer this conflict refers to requirement 4 from Figure 3.3. For conflict 1.1 there are two resolution option, 89-A and 89-B, which are described as "HTML editor integrated into system" and "External HTML editor" respectively. The next four columns are the Client's contribution to the QAScores followed by the calculated benefit for the Client if a particular resolution option was

chosen. The Developer's contribution to the QAScores and benefit is next. This is followed by the relative cost of the resolution options, which in this case is 50 for 89-A and 10 for 89-B. In the next column is the calculated mean desirability of each option. The option with the higher mean desirability is the "winner," which for conflict 1.1 is 89-B. This "winner" was a clear winner because it's mean desirability is greater than one compared to the mean desirability of the other resolution option. 89-B was also the resolution adopted by the model-free negotiators.

First Win-Win Iteration

The conflicts that we compiled for the first iteration are represented in Figure 3.14 and Figure 3.15. In the first iteration, out of a 112 issues compiled by the stakeholder groups, the investigator identified 21 requirements conflicts that needed to be resolved. The first 10 conflicts are in Figure 3.14 while the remaining are in Figure 3.15.

The first conflict, 1.1, was the type of HTML editor to use. Solution A is to have an HTML editor integrated into the system. This option was heavily favored by the client and greatly contributes to three of the clients objectives explaining the +1 contributions. The developer did not want to develop an HTML editor and it somewhat negatively contributes to two of the developer's objectives resulting in two -0.5 ratings. Solution B involves creating templates using an external HTML editor and uploading them later into the system. This solution positively contributes to two

A	Ability to distribute emails with video streams to contact lists						Client QAScore					Developer QAScore															
B	Simple interface designed for effective use						A	B	C	D		A	B	C	D												
C	Ability to send emails for advertising purposes						30	30	25	15		50	15	15	20												
D	Accountability of activities performed in order to prevent misuse																										
	Conflict		Win Condition	Resolution	Resolution		Client Contribution				Client	Developer Contribution				Developer	Cost	Mean								
#	Description		Client	Develop	Option	Description	A	B	C	D	Benefit	A	B	C	D	Benefit		Desirability									
	1.1	product has no interface to create templates	5	5	5-A	HTML editor integrated into system	1	1	1	0	85	-0.5	-0.5	0	0	-32.5	50	0.525									
					5-B	External HTML editor	1	-1	1	0	25	1	0.5	0	0	57.5	10	4.125	winner clear		m-f						
	1.2	manager should preview exact e-mail as customer not sample	13	7	11-A	manager only previews e-mails	0	0	0	1	15	0	0	0	0	0	10	0.750			m-f						
					11-B	manager can edit e-mails	0	0	0	0.5	7.5	0	0	0	1	20	10	1.375	winner marginal								
	1.3	resolution of e-maill distribution collisions	12	10	16-A	Scheduling system	1	0	1	0.5	62.5	-1	0	-1	0	-65	80	-0.016									
					16-B	collisions queued	0.5	0	0.5	0	27.5	0	0	0	0	0	20	0.688	winner marginal								
					16-C	collisions abondoned	-0.5	0	-0.5	0	-27.5	0.5	0	0.5	0	32.5	10	0.250									
	1.4	cascading access privileges	8	13	35-A	include cascading access	0	0	0	1	15	0	0	0	-0.5	-10	20	0.125	winner marginal		m-f						
					35-B	exclude cascading access	0	0	0	0	0	0	0	0	0	0	20	0.000									
	1.5	consumer database and target list is separate but changes will be made to both	3	4	66-A	target list separate from consumer list	1	0	0	0	30	-0.5	0	0	0	-25	30	0.083333	winner marginal		m-f						
					66-B	no target list included	-1	0	0	0	-30	0.5	0	0	0	25	10	-0.25									
	1.6	product has no ability to format content	10	5	43-A	content editor built into system	1	1	0.5	0	72.5	-0.5	-0.5	0	0	-32.5	50	0.4	winner clear		m-f						
					43-B	no content editor	-0.5	-1	0	0	-45	0.5	0.5	0	0	32.5	10	-0.625									
	1.7	automatic notification of acceptance or rejection of e-mail to both employee and manager	6,13	7	46-A	add automatic notification	0	0	0.5	1	27.5	0	0	0	-0.5	-10	20	0.4375	winner clear		m-f						
					46-B	dont add automatic notification	0	0	-0.5	-1	-27.5	0	0	0	0	0	10	-1.375									
	1.8	must have and easy to update help section	9	14	55-A	make the help system in wiki	0	0.5	0	-0.5	7.5	0	1	0	0	15	10	1.125	winner marginal								
					55-B	built-in a help system as originally planned	0	1	0	1	45	0	-0.5	0	0	-7.5	30	0.625									
	1.9	need to be able to manage the links between template and media	10	-	62-A	media is attached to template	1	0	1	0	55	1	0	1	0	65	40	1.5	winner marginal								
					62-B	media is URL link to off-site servers	0.8	0	1	0	49	-0.5	0	1	0	-10	20	0.975									
	1.10	what kind of information is on the consumer lists	3	4	65-A	variable fields in text files	1	0	0	0	30	-0.5	0	0	0	-25	20	0.125									
					65-B	pre-defined fields in fixed format files	0.5	0	0	0	15	1	0	0	0	50	10	3.25	winner clear								

Figure 3.14: First Win-Win Iteration Part 1, for the Application of the Win-Win Model

objectives but negatively contributes to the simple interface objective. The developer prefers this solution to avoid writing a new HTML editor, and this solution positively contributes to two developer objectives. The cost of solution A is 50 compared to 10 for solution B because developing an HTML editor is a far more time-consuming than developing an uploading of template solution. Solution B had the highest desirability by a clear margin over solution A.

Conflict 1.2 has two solutions, which are A. manager previews e-mails or B. manager previews and edits e-mail. Both resolutions are equally easy to implement, therefore they both cost 10. Resolution A greatly contributes to the client's accountability objective, thus receiving 1, while resolution B somewhat contributes to the objective so receiving 0.5. Resolution A does not affect the developer, but resolution B greatly contributes to the developer's accountability objective. Resolution B was the winner marginally.

For conflict 1.3 there were three resolutions to solve the e-mail collisions problem. Option A was creating a scheduling system, which was greatly desired by the client but not by the developer because of the time involved in developing it. This option had a heavy cost of 80. Option B cost a lot less at 20 and was somewhat desired by the client but the developer is neutral to the idea. Option C cost even less at 10 but where undesirable to the client because e-mails could be lost due to abandonment. The developer is somewhat favored option C as it contributes 0.5 to two developer

objectives. Option B, the queuing of collisions, was the marginal winner.

Conflict 1.4 had to deal with the issue of including cascading access to users. Resolution A was to implement as getting access which was favored by the client for accountability purposes but not desired by the developer because of the extra work needed. Both stakeholders were neutral to resolution B. Both resolutions equally low in cost at 20. Resolution A, the cascading of privileges, was the marginal winner.

Conflict 1.5 address the issue of whether creating a separate customer target list from the main consumer list. The main problem with having separate lists is propagating change from one list to another. Option A which was to include a separate target list, which was greatly favored by the client but somewhat undesirable for the developer. Option B calls for taking no action, which is favored by the developer let not the client. Option A was the marginal winner of this conflict.

Conflict 1.6 was whether to include a content or media editor into the system. This conflict was based more fundamental question of whether content editing was needed. The client heavily favored including the content editor, resulting in positive marks for resolution A and negative marks for resolution B, while the developer was not keen to develop the content editor and is reflected in the chart. Including a content is far more costly than having no content editor therefore the content editor costs 50 while not doing so costs 10. The option to build a content editor was the clear winner.

In conflict 1.7, whether to add automatic notification was debated. The client desired automatic notification or solution A while the developer was only somewhat against it. The client was heavily against solution B, not including automatic notification, while the developer was neutral to this solution. Adding automatic notification was only slightly more costly than the other solution resulting in option A costing 20 and option B 10. Option A, adding automatic notification, was the optimal solution.

Conflict 1.8 addressed the nature of the help section required by the system. One option was to have a wiki while the other option was to have a traditional help section. The client was unsure about the wiki therefore had a 0.5 rating for the user interface objective but -0.5 rating for the accountability objective because they're unsure how accountable a wiki would be. The developer was in favor of the wiki especially for objective B. The client was heavily in favor of the traditional help, while the developer was somewhat against it. The traditional help was costlier than the wiki three times more. The option to use a wiki for the help system was the winner, marginally.

Conflict 1.9 dealt with managing links between template and media. One option was to attach the media to the templates which was strongly favored by both stakeholders despite costing twice as much as the other option. Option B was to have the templates link to the media on a server which was favored by the client but not as much by the developer because they would've had to write a lot of code. The option to attach media to the template had the highest desirability, marginally.

Conflict 1.10 had to do with storing contact information, where the option A was to have the contact information stored in variable fields and option B list of predefined fields. Option A cost slightly more than option B where they cost 20 and 10 respectively. Option A was greatly desired by the client and was somewhat undesired by the developer. Option B wasn't greatly favored by the developer and somewhat favored by the client. Option B was the clear winner of this conflict.

Conflict 1.11 from Figure 3.15 was a debate about addressing the import of new contacts. Option A overwriting data with manual confirmation while option B was to overwrite data without confirmation. Both options cost relatively little but the manual confirmation requires a little more work. The client favored manual confirmation while the developer did not which is reflected in the numbers. The option to simply overwrite contacts was the clear winner.

Conflict 1.12 was whether or not to create target lists from performance reports. Option A, adding the functionality, is costlier that option B, not adding the functionality. The numbers are reflected by the fact the client favored having this functionality while the developer did not. Option A was the clear winner of this conflict.

During one of the videoconference session the client proposed a new requirement to report unsubscriptions from the service leading to conflict 1.13. The first option was to fully report unsubscribing customer which was heavily and desired by the client and heavily undesired by the developer. This option cost the most at 80. The

A	Ability to distribute emails with video streams to contact lists						Client QAScore						Developer QAScore																		
B	Simple interface designed for effective use						A	B	C	D			A	B	C	D															
C	Ability to send emails for advertising purposes						30	30	25	15			50	15	15	20															
D	Accountability of activities performed in order to prevent misuse																														
	Conflict	Win Condition		Resolution	Resolution		Client Contribution				Client		Developer Contribution				Developer	Cost	Mean												
#	Description	Client	Develo	Option	Description		A	B	C	D	Benefit		A	B	C	D	Benefit		Desirability												
1.11	how does system deal with	3	4	89-A	manual confirmation of overwriting		1	0	0	0	30		-0.5	0	0	0	-25	15	0.166667												
	importing new contacts			89-B	updating is overwriting		-0.5	0	0	0	-15		1	0	0	0	50	5	3.5	winner	clear										
1.12	no functionality to create target list	3	4	90-A	target list function added		0.5	0	1	0	40		-0.5	0	-0.5	0	-32.5	25	0.15	winner	clear	m-f									
	from reports or manually			90-B	functionality not added		0	0	-1	0	-25		0	0	0.5	0	7.5	10	-0.875												
1.13	requires functionality to report	0	0	93-A	full reporting on unsubscription		1	0	0	1	45		-1	0	0	-1	-70	80	-0.15625												
	unsubscription over time interval			93-B	unsubscriptions tracked as reporting metric		1	0	0	1	45		-0.5	0	0	-0.5	-35	40	0.125	winner	marginal	m-f									
				93-C	unsubscriptions untracked		0	0	0	-0.5	-7.5		0	0	0	0	0	10	-0.375												
1.14	all e-mail distribution should be	0	0	96-A	add such a database		0	0	0	1	15		0	0	0	-0.5	-10	40	0.0625		marginal	m-f									
	tracked and stored in a database			96-B	requirement not added		0	0	0	-0.5	-7.5		0	0	0	0.5	10	10	0.125	winner											
1.15	what is the structure of the statistic	7	11	98-A	manually create structure of report		0	1	0	1	45		0	-0.5	0	-1	-27.5	40	0.21875												
	reports			98-B	have a predefined report structure		0	0.5	0	0.5	22.5		0	0.5	0	0.5	17.5	10	2	winner	clear										
1.16	automatically find the right approval	6,13	8	110-A	approval manager is predefined		0	0.5	0	0.5	22.5		0	0	0	-0.5	-10	25	0.25	winner	marginal										
	manager			110-B	manually submit to approval manager		0	-0.5	0	-0.5	-22.5		0	0	0	0.5	10	10	-0.625												
1.17	who initially schedules e-mail	0	0	3-A	marketing employees schedule e-mails		0.5	0	0	0	15		0	0	0	0	0	10	0.75	winner	clear	m-f									
	blasts			3-B	marketing managers schedule e-mails		-0.5	0	0	0	-15		0	0	0	0	0	10	-0.75												
1.18	functionality needed to deal with	3	3	8-A	bounced e-mail addresses removed		-0.5	0	0	0	-15		0.5	0	0	0	25	10	0.5	winner	marginal										
	bounced e-mails			8-B	bounced e-mail addresses are dealt with		1	0	0	0	30		-0.5	0	0	0	-25	20	0.125			m-f									
1.19	client requires instant e-mail	12	10	15-A	requirement added		0.5	0	1	0	40		-0.5	0	0	0	-25	15	0.5	winner	marginal	m-f									
	distribution, rather than just			15-B	requirement not added		0	0	-0.5	0	-12.5		0.5	0	0	0	25	10	0.625												
	scheduled																														
1.20	system must be compatible with	14	0	29-A	compatibility added		0	0.5	0	0	15		0	-0.5	0	0	-7.5	15	0.25			m-f									
	Mozilla Firefox and Linux			29-B	compatibility not added		0	0	0	0	0		0	0.5	0	0	7.5	10	0.375	winner	marginal										
1.21	how are the statistic reports	7	11	49-A	reports are scheduled		0	0	0	0.5	7.5		0	0	0	0.5	10	10	0.875	winner	marginal										
	scheduled			49-B	reports are scheduled and ad-hoc		0	0	0	1	15		0	0	0	-0.5	-10	20	0.125												

Figure 3.15: First Win-Win Iteration Part 2, for the Application of the Win-Win Model

second option was to track unsubscribing customers in the performance reports which was also heavily favored by the clients but somewhat undesirable for the developer. This option cost 40 because it was less than the first option. The third option was not including this requirement which was somewhat unfavored by the client and neutral to the developer. This option had a cost rating of 10 because there are no implementation costs. The option to track unsubscriptions as a performance reporting metric was the winner.

Conflict 1.14 was the tracking the e-mail distributions in a database. Creating a database was far more costly than not implementing this requirement by 40 to 10. The client was in favor of this requirement therefore receiving a positive rating for one of their objectives, while the developer do not desire this requirement. Not including this requirement was somewhat undesired by the client receiving a -0.5 rating, while the developer would have been somewhat pleased receiving a +0.5 rating. It was optimal not include this database in the system, so the requirement wasn't added.

Conflict 1.15 address the structure of the performance reports. One solution was to have a customizable structure which cost more than the other solution, which was to have predefined report structure. Both solutions had a positive impact on the client but the client favored the customizable structure more. The developer favored that predefined report structure rather than the customizable structure because this option required more coding. The option to have predefined report structure was the

winner.

Conflict 1.16 was how the approval manager is selected once an employee creates an e-mail. Solution A is to have the system select the approval manager based on predefined criteria, while solution B is having the employee select the manager manually. The client favored solution A while the developer did not. Solution B negatively impacted the client while having a positive impact on the developer. The first solution was slightly more costly than the second solution. The option to have the approval managers predefined was the winner.

Conflict 1.17 was brought by the client after releasing their initial requirements set. The issue was who initiates e-mail campaigns. There were two solutions that cost nothing to implement. The clients favored solution A but not solution B. The developer was neutral to both solution. Solution A was a marginal winner.

This issue, conflict 1.18, was based on how to deal with bounced e-mail addresses. One solution is to immediately delete bounced e-mail addresses from the contact lists, while the other costlier solution was to "deal with" such addresses by resending to these addresses before deleting. The first solution was favored by the developer was not the client. The second solution was greatly favored by the client was not the developer. The option to promptly remove bounced e-mail addresses was chosen as the winner.

For Conflict 1.19, there were two solutions, whether to add the requirement or

not. Implementing the requirement costs very little at 15. The rating numbers reflect the fact that the clients favored this requirement while the developer did not desire this requirement. This requirement was added.

Conflict 1.20 was based on the fact that the client wanted the system to be compatible with Mozilla and Linux as well as Internet Explorer and Windows. Adding this requirement and did not cost much more at 15. The client favored this new requirement while the developer was not very pleased with this development. The winning option was not to add the extra compatibility.

Conflict 1.21 addressed the scheduling of performance reports. Option A was to have the reports produced on a regular basis while Option B, which is slightly more expensive, was to have the reports regularly scheduled and previous ad hoc. The client favored both options tilting towards option B more. The Developer favored option A but not option B. Option A, having only scheduled reports, was the winner.

Observations

For the first iteration, we can immediately observe that CBAM was able to find the most desirable solution for each of the 21 conflicts. In eight of the conflicts, the solutions found were clear "winners" and the remaining 13 of the solutions were "marginal" winners.

In the model-free version of these negotiations 13 conflicts were resolved, 8 conflicts were not, because the negotiators could not come to an agreement. Two of the 13 of

the that were resolved had to be reopened for discussion in later sessions. Only one issue clearly resolved by CBAM had to be reopened in the model-free analysis.

Second Win-Win Iteration

In the second iteration, Figure 3.16, eight conflicts not resolved in the initial model-free iteration were reevaluated with the CBAM. These conflicts had already been solved by the win-win model in the first iteration, but we are using new resolution options for these conflicts being reevaluated.

Conflict 2.1 of Figure 3.16 is a revisitation of conflict 1.3 of Figure 3.14 regarding collision of e-mails on the server. In the first iteration it was decided to queue colliding e-mails, but the stakeholders were debating whether to add notification of each collision. Adding notification was desired by the client with a 0.5 rating, while it was not desirable for the developer giving a rating of -0.5. The client was neutral if notification was not included while the developer seemed to be happy with the exclusion of this requirement. Adding notification was only slightly costy at 15. It was decided not to include e-mail notification.

Conflict 2.2 is a continuation of conflict 1.8 regarding the help system. It was decided to use a wiki in the first iteration, but the second time the stakeholders and discuss the nature of the wiki. Option C was to have a completely open wiki costing a little bit at 25, while option D is to have a moderated wiki which cost more at 40. The

A	Ability to distribute emails with video streams to contact lists						Client QAScore					Developer QAScore																	
B	Simple interface designed for effective use						A	B	C	D		A	B	C	D														
C	Ability to send emails for advertising purposes						30	30	25	15		50	15	15	20														
D	Accountability of activities performed in order to prevent misuse																												
	Conflict	Win Condition		Resolution			Client Contribution				Client	Developer Contribution				Developer	Cost	Mean										
	Description	Client	Develop	Option	Description		A	B	C	D	Benefit	A	B	C	D	Benefit		Desirability											
2.1	resolution of e-maill distribution collisions	12	10	16-D	collisions queued w/ e-mail notification		0	0	0	0.5	7.5	0	0	0	-0.5	-10	15	-0.083											
				16-E	collisions queued w/o e-mail notification		0	0	0	0	0	0	0	0	0.5	10	10	0.500	winner	marginal	m-f									
2.2	must have and easy to update help section	9	14	55-C	have a "live" wiki help system		0	0	0	-0.5	-7.5	0	0	0	0.5	10	25	0.05											
				55-D	wiki help system with moderator		0	0.5	0	1	30	0	0	0	-0.5	-10	40	0.25	winner	marginal	m-f									
2.3	need to be able to manage the links between template and media	10	-	62-C	use search function to retrieve URL		0.5	0	0	0	15	1	0	0	0	50	10	3.25	winner	clear	m-f								
				62-D	create a URL management system		0.5	0	0.5	0	27.5	-0.5	0	-0.5	0	-32.5	30	-0.0833333											
2.4	what kind of information is on the consumer lists	3	4	65-C	data must be in varying formats in text files		1	0	0	0	30	-0.5	0	0	0	-25	50	0.05			m-f								
				65-D	data is fixed in text files		0.5	0	0	0	15	0.25	0	0	0	12.5	20	0.6875	winner	marginal										
2.5	how does system deal with importing new contacts	3	4	89-C	synchronize addresses when uploading new		1	0	0	0	30	-0.5	0	0	0	-25	30	0.0833333	winner	marginal	m-f								
				89-D	overwrite addresses when uploading new		-1	0	0	0	-30	0.5	0	0	0	25	10	-0.25											
2.6	what is the structure of the statistic reports	7	11	98-A	manually create structure of report		0	1	0	1	45	0	-0.5	0	-1	-27.5	40	0.21875											
				98-B	requirement not added		0	0.5	0	0.5	22.5	0	0.5	0	0.5	17.5	10	2	winner	clear									
2.7	automatically find the right approval manager	6,13	8	110-C	campaign manager added to assign approvals		0	0	0	1	15	0	0	0	0	0	15	0.5	winner	marginal	m-f								
				110-D	campaign manager not added		0	0	0	-0.5	-7.5	0	0	0	0	0	10	-0.375											
2.8	how are the statistic reports scheduled	7	11	49-A	reports are scheduled		0	0	0	0.5	7.5	0	0	0	0.5	10	10	0.875	winner	marginal									
				49-B	reports are scheduled and ad-hoc		0	0	0	1	15	0	0	0	-0.5	-10	20	0.125											

Figure 3.16: Second Win-Win Iteration, for the Application of the Win-Win Model

developer favored the open wiki while the client did not. The client heavily favored the moderated wiki which was disapproved by the developer. Option D, having a moderator, was the optimal solution.

Conflict 2.3 is a continuation of conflict 1.9 addressing the management of media. They discussed whether search for links to the media or have the links inserted in the e-mails requiring a URL management system with a somewhat considerable cost that we rated 30. The search capability costs nothing. The numbers reflect the fact that the client was in favor of both solutions with a slightly more emphasis on the URL management. The developer favored the search capability and did not desire the URL management system. The option to use a search function to retrieve the URL was the clear winner.

Conflict 2.4 is a revisitation of conflict 1.10. The resolution options are the same as in the first iteration and so are the contribution values. Once again the second option was the winner.

Conflict 2.5 is the continuation of conflict 1.11 regarding the importing of new contacts. There were two solutions to this conflict, one was to synchronize the new addresses which is costing 30 or the other option was to simply overwrite the new addresses which cost nothing. The contribution numbers reflect the fact that the client was in favor of synchronizing addresses while the developer was in favor of overwriting the new addresses. The option to synchronize addresses when they're

uploaded was the winner of this conflict.

Conflict 2.6 is a revisitation of conflict 1.15. The resolution options are the same in this conflict as they were in conflict 1.15. Again it was decided to have predefined report structure rather than a customizable report structure.

Conflict 2.7 was a continuation of conflict 1.16 regarding the locating of an approval manager. The issue was whether to create a campaign manager who will assign approval manager. The addition of a campaign manager costs very little and we rated it 15. The client was in favor of adding a campaign manager while the developer was neutral to the idea. Adding the campaign manager is a good requirement according to the model.

Conflict 2.8 is a revisitation of conflict 1.21. The resolution options are the same as in the first iteration and so are the contribution values. The first option was again the winner.

Observations

Our immediate observation is that CBAM was able to again find the most desirable solution for every single conflict. two of the solutions were clear "winners" and six solutions were only "marginal" winners. In the model-free version of this iteration, six out of eight issues were resolved and two out of eight were unresolved and were never reopen for discussion.

Third Win-Win Iteration

In the third and final iteration, two conflicts needed to be solved, see Figure 3.17. Both of these conflicts were issues that were solved in the first iteration, but needed to be reopened for discussion.

Conflict 3.1 of Figure 3.17 is the reopening of conflict 1.17 involving the initiation of e-mail campaign. It was initially decided to allow only employees to schedule e-mails but the client changed their mind and demanded that the managers initialize e-mail campaigns. Both solutions had no cost associated with it receiving a rating of 10 each. The developer was in favor of employees because they wouldn't have to change the documentation. The option to allow marketing managers to schedule e-mail was the clear winner.

Conflict 3.2 is a clash between the solution of conflict 1.2 and 1.4, where the decision to prevent managers from editing e-mails in 1.2 was conflicting with the decision in 1.4 to have cascading privileges where the manager has all the privileges of an employee plus new privileges. The first option, which does not cost, was to drop cascading privileges favored by the client. The second option, which cost slightly more at 15, is to allow cascading privileges favored by the developer. The second option, to allow cascading privileges, was the winner in this conflict.

Observations

We observe that CBAM was able to find the best solution for both conflicts, but

		Client QAScore				Developer QAScore			
		A	B	C	D	A	B	C	D
A	Ability to distribute emails with video streams to contact lists								
B	Simple interface designed for effective use								
C	Ability to send emails for advertising purposes	30	30	25	15	50	15	15	20
D	Accountability of activities performed in order to prevent misuse								

#	Conflict	Win Condition		Resolution	Resolution	Client Contribution				Client	Developer Contribution				Developer	Cost	Mean			
	Description	Client	Develop	Option	Description	A	B	C	D	Benefit	A	B	C	D	Benefit		Desirability			
3.1	client requires managers to			3-A	marketing employees schedule e-mails	-0.5	0	0	-0.5	-22.5	0	0	0	0.5	10	10	-0.625			
	schedule e-mail blasts not			3-B	marketing managers schedule e-mails	0.5	0	0	0.5	22.5	0	0	0	0	0	10	1.125	winner	clear	m-f
	employees changing issue 1.17																			
3.2	since manager only approves	13	7	11-A	manager only previews e-mails/no cascading priv	0	0	0	1	15	0	0	0	0	0	10	0.750			m-f
	and not creates e-mails, cannot			11-B	manager can edit e-mails/cascading privileges	0	0	0	0.5	7.5	0	0	0	1	20	15	0.917	winner	marginal	
	have cascading privileges																			

Figure 3.17: Third Win-Win Iteration, for the Application of the Win-Win Model

it was solved clearly for one issue but only marginally for the other. In the model-free negotiations both issues were resolved.

3.5 Application of *i** Framework

The procedure of the application will be presented first followed by the results.

3.5.1 Procedure

Visualizations of the conceptual software system were created after every negotiation session from each stakeholder team's viewpoint. These visualizations were made, based on the Strategic Dependency (SD) [34] model, under which the ***i**** Framework operates. As discussed in Section 2.3.3 SD is the model that treats every actor as a single unit and has dependencies with other actors. Unfortunately, by using SD, only actor-based conflicts can be addressed, and no other conflict was addressed.

The users[3] of the system and their roles were initially defined in the Request for Proposal (RFP) document written by the Client. This point of view was captured in the first visualization, Figure 3.18. Both the Client and the Developer brought in two conflicting views in the 1st round of negotiations (Requirements Elicitation session). These views were captured in two separate visualizations, Figure 3.19 and

[3]User or actor refers to the end-user who will be utilizing the system being negotiated. Several types of users have been identified, such as employee and manager.

Figure 3.5.2. The next visualization, Figure 3.21, was based on the first Requirement Specification (RS1), which was the Developer's take on what was developed in the previous session. In the next step, visualizations, Figure 3.22 and Figure 3.23, were created for both sides coming into the 2nd round of negotiations (Requirements Negotiation session). Another couple visualizations, Figure 3.24 and Figure 3.25, were made for the third round of negotiations (Prototype Demonstration session). A final visualization, Figure 3.26, was produced to capture the Developers point of view as expressed in the second Requirement Specification document (RS2). This image was the final requirements configuration on which it was agreed.

3.5.2 Results

Figure 3.18 to Figure 3.26 are the *i** framework visualizations at different stages of the requirements negotiation process in the case study.

Request for Proposal (RFP)

The Request for Proposal (RFP) was the first document produced by the Client. In the RFP document the roles for the different human users was very vague. The only manual user tasks required at this stage were: someone to manually update contact lists in addition to an automated contact list updater (requirements 3 in Figure 3.2); an ability for users to preview e-mails before distribution (requirements

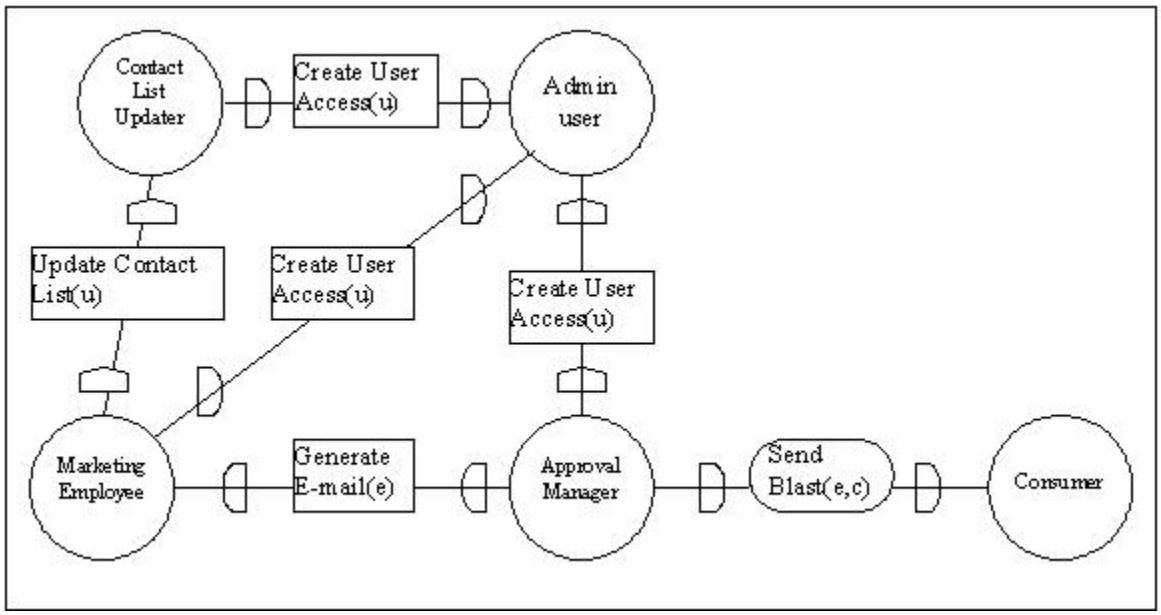

Figure 3.18: Client View extracted from the RFP, in the Application of the ***i**** framework

6 in Figure 3.2); an administrative user level who controls and creates user access to the software solution – they also decide other user access levels and their respective job function (requirements 8 in Figure 3.2); and an approval managers who preview and approve any e-mail being distributed (requirements 10 in Figure 3.2). Three of the user roles, contact list updater, approval manager and employee, were easily understandable, but the administrator's function was a little vague and needed further work for the Developer. The Consumer was presumed to be another actor in the system. Figure 3.18 is the a visualization of the role of the users at this stage.

Requirements Elicitation Session

The users and their roles were further defined in the requirements elicitation session and appeared in the first Requirement Specification. The following users and actors were defined: (1) Administrative level (2) Internal Marketing Users (3) External Marketing Users; (4) Maintainer. The top administrative level decides what each level is allowed to do – the Client wanted it to be customizable by the administrator.

The Maintainer Level is able to do debugging if there is a system problem. The Developer discovered that the Maintainer Level in the product does not have the same privileges as the administrative level and believed the Maintainer Level should have full access. The Client believed that access should be limited because they don't want the Developer to have unauthorized uses of the product. In the Developer view

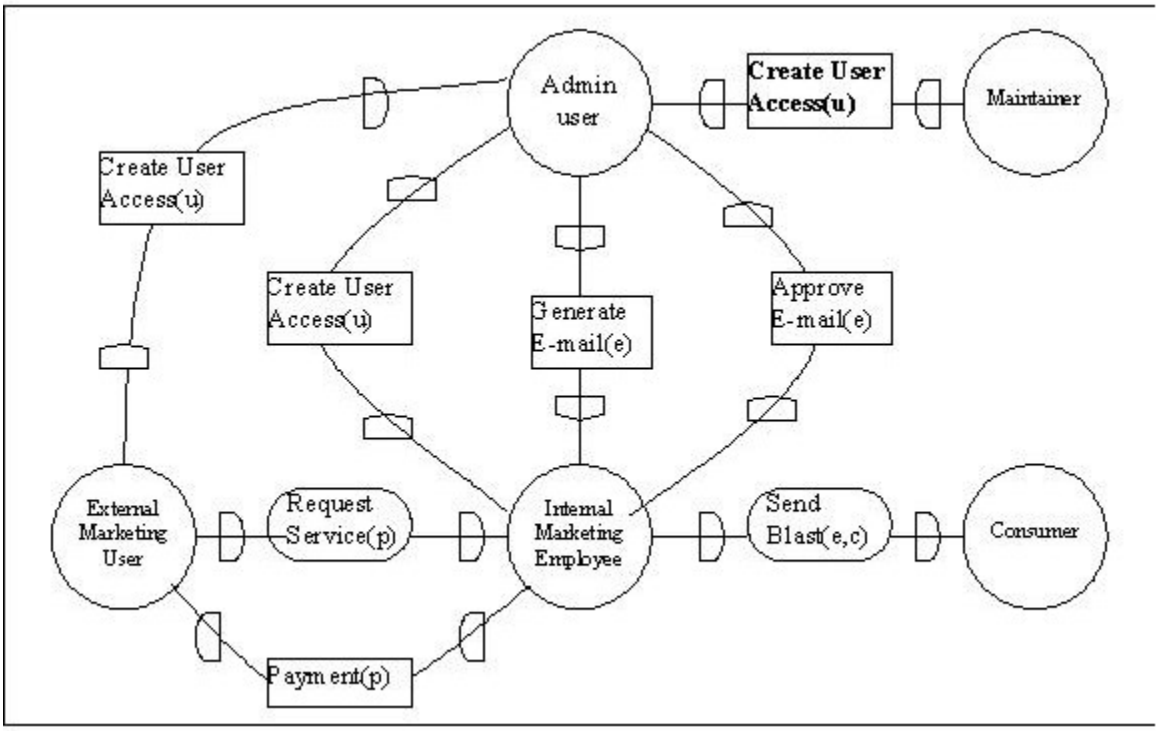

Figure 3.19: Clients' View extracted from the Requirements Elicitation Session, in the Application of the ***i**** framework

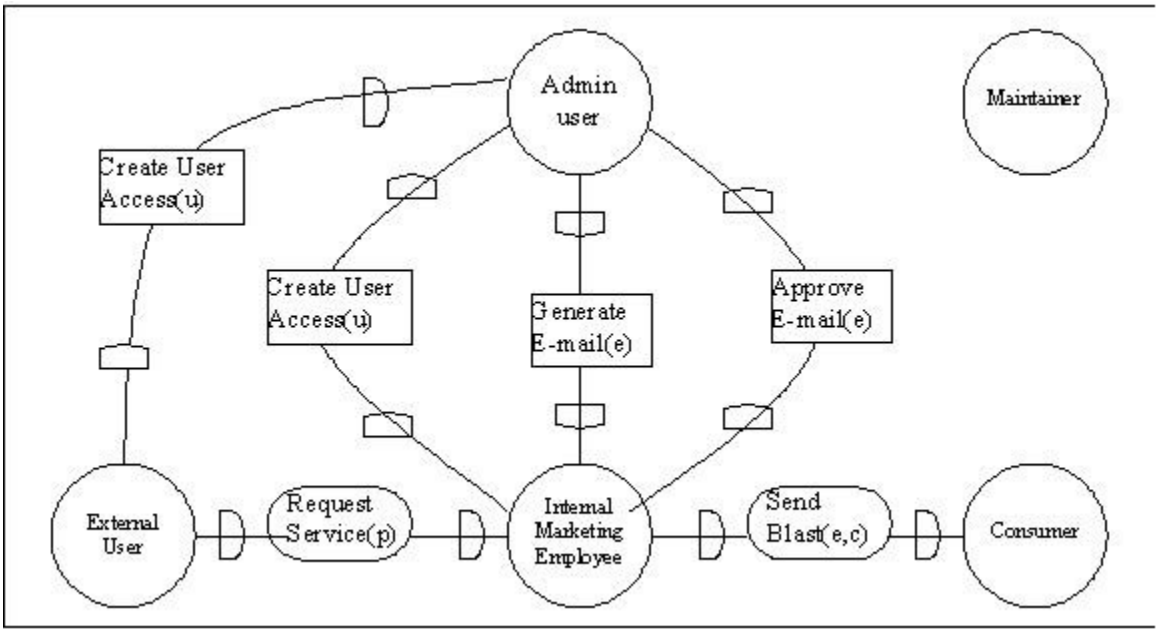

Figure 3.20: Developers' View extracted from the Requirements Elicitation Session, in the Application of the ***i**** framework

of Figure 3.5.2 one can see that the Maintainer Level does not require access from the Administrator, while in Figure 3.19 the Client envisions the Maintainer Level requires access. The Developer basically accepting the clients view to resolve this conflict.

Another conflict raised was the use of performance metrics of the system and how the users of the system will employ the metrics delivered. The result of that was a creation of the executive personnel who is main interest is to analyze these metrics. The executive personnel were added in the Requirements Specification version 1.0.

There was an addition of an External User that was added by the client, as one can see the difference between Figure 3.18 and Figure 3.5.2. The developer also envisioned An External User with the same responsibilities; therefore conflict was avoided in this case.

The character that updates the contact list was cut back, and that role was merged into the internal marketing employee.

Requirement Specification Version 1.0

The first Requirement Specification document shows the results of the requirements elicitation session. Figure 3.21 shows the resolved view of the previous session. There were five users' roles that were explicitly defined in the first Requirement Specification document. These users were: executive personnel, maintainers/testers, marketing managers, marketing employees, and business partners. Consumers, while not ex-

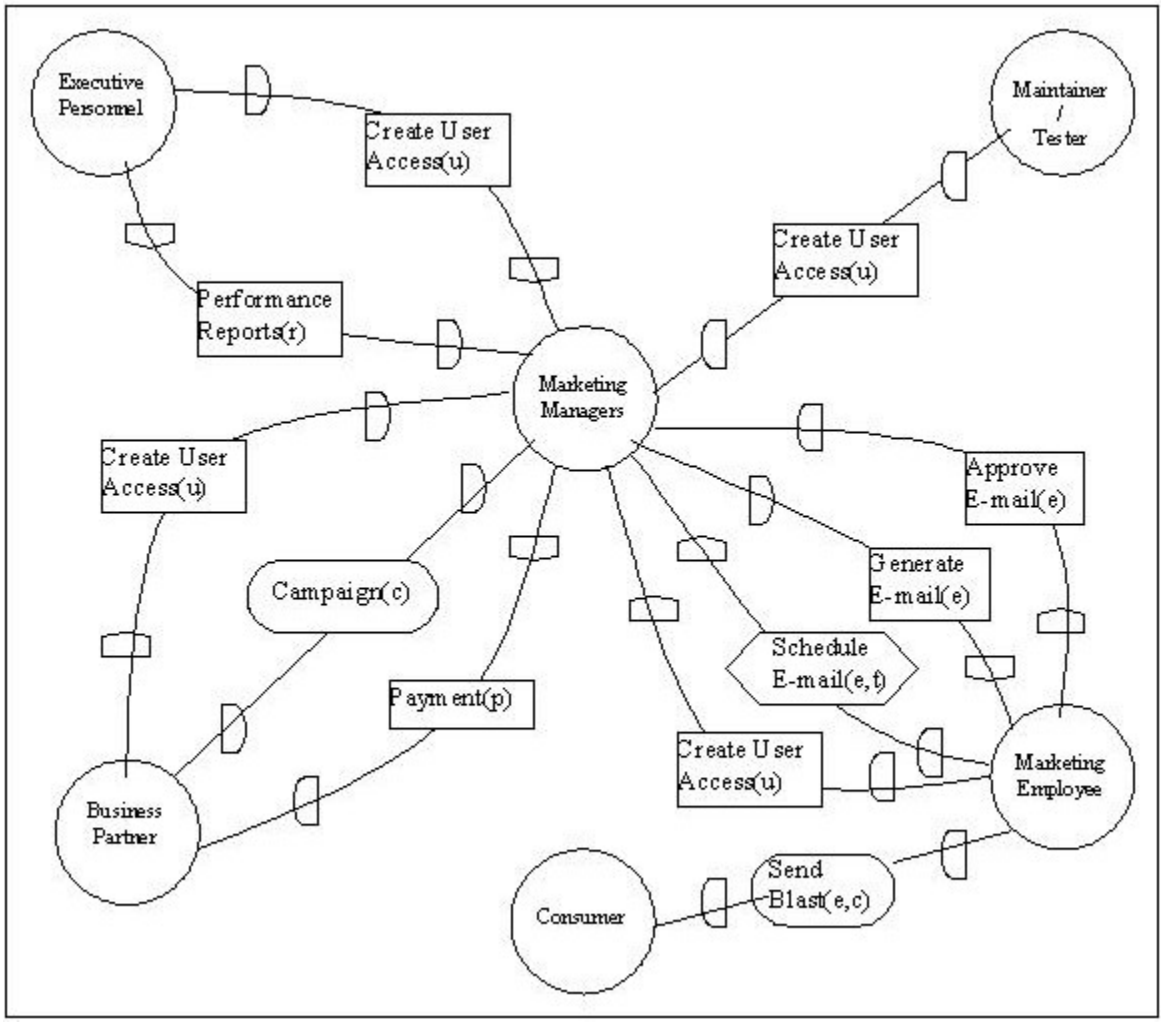

Figure 3.21: Developers' View extracted from the Requirements Specification version 1.0, in the Application of the ***i**** framework

plicitly stated in the document, were added to the visualization. These users remain unchanged for the rest of the process.

Requirement Specification Inspection and Requirements Negotiations

During the inspection and negotiation process two issues arose that relate to role of the user. The first issue was whether a Manager who approves e-mails should be able to edit the e-mails or not. The Developer was confused on this issue, but the Client reiterated that a Manager should see an e-mail the way the consumer would see it and the Manager only accept or reject an e-mail for distribution. The Manager had no editing powers. The conflict was resolved by simply accepting the Clients view. In another minor issue, it was resolved that the system automatically sends an e-mail for approval to the right manager.

It was decided that a Campaign manager was to be added to direct new advertising campaigns. Campaign manager was to be defined by the developer. Campaign manager was never defined in the documentation, so it was added as someone with the same responsibilities as a manager in the resolved view.

Prototype Demonstration

The prototype demonstration was the last presentation by the Developer of the system to the Client, before the final requirement specification document was delivered. As a

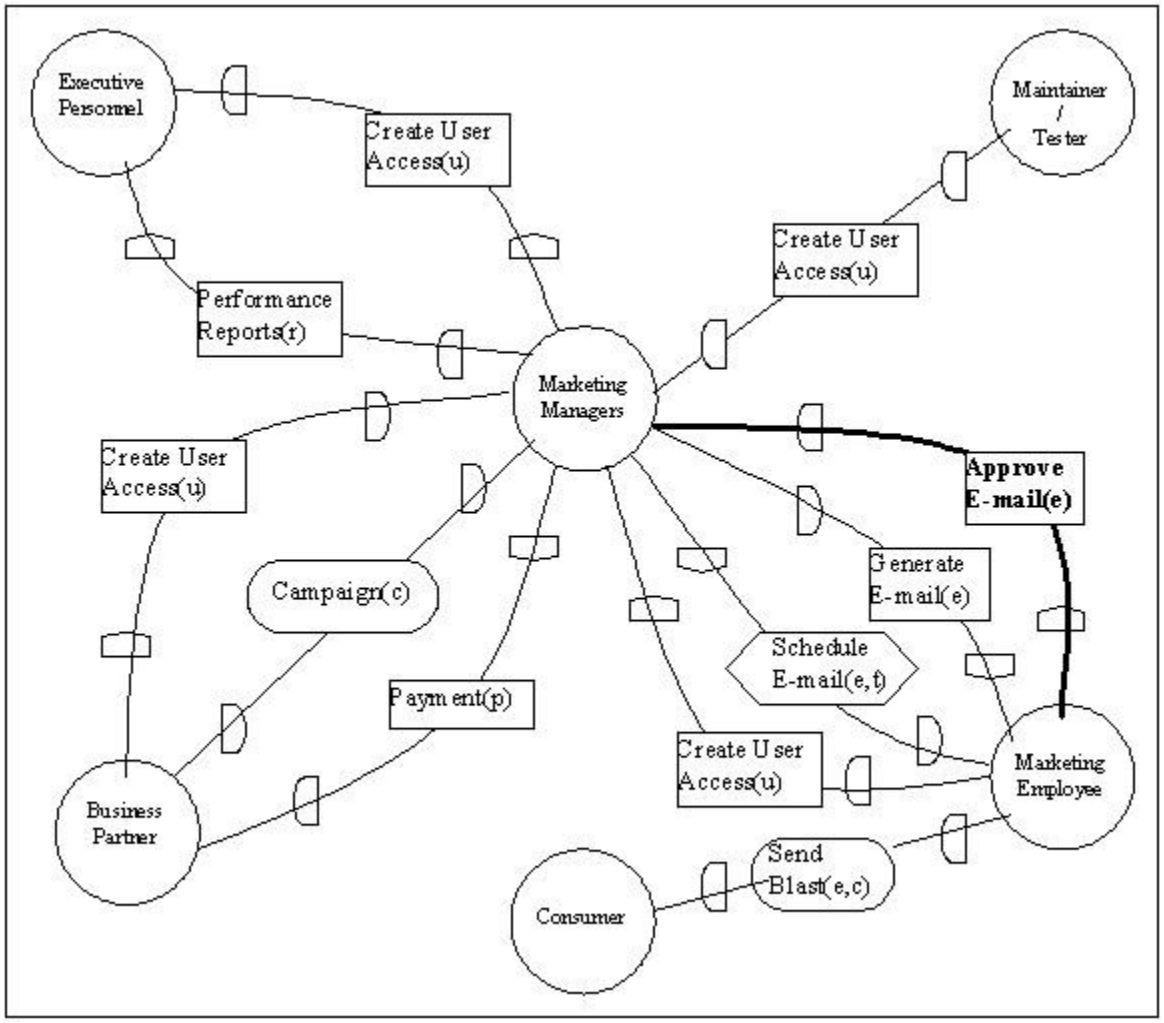

Figure 3.22: Requirements Negotiations Clients' View extracted from the , in the Application of the ***i**** framework

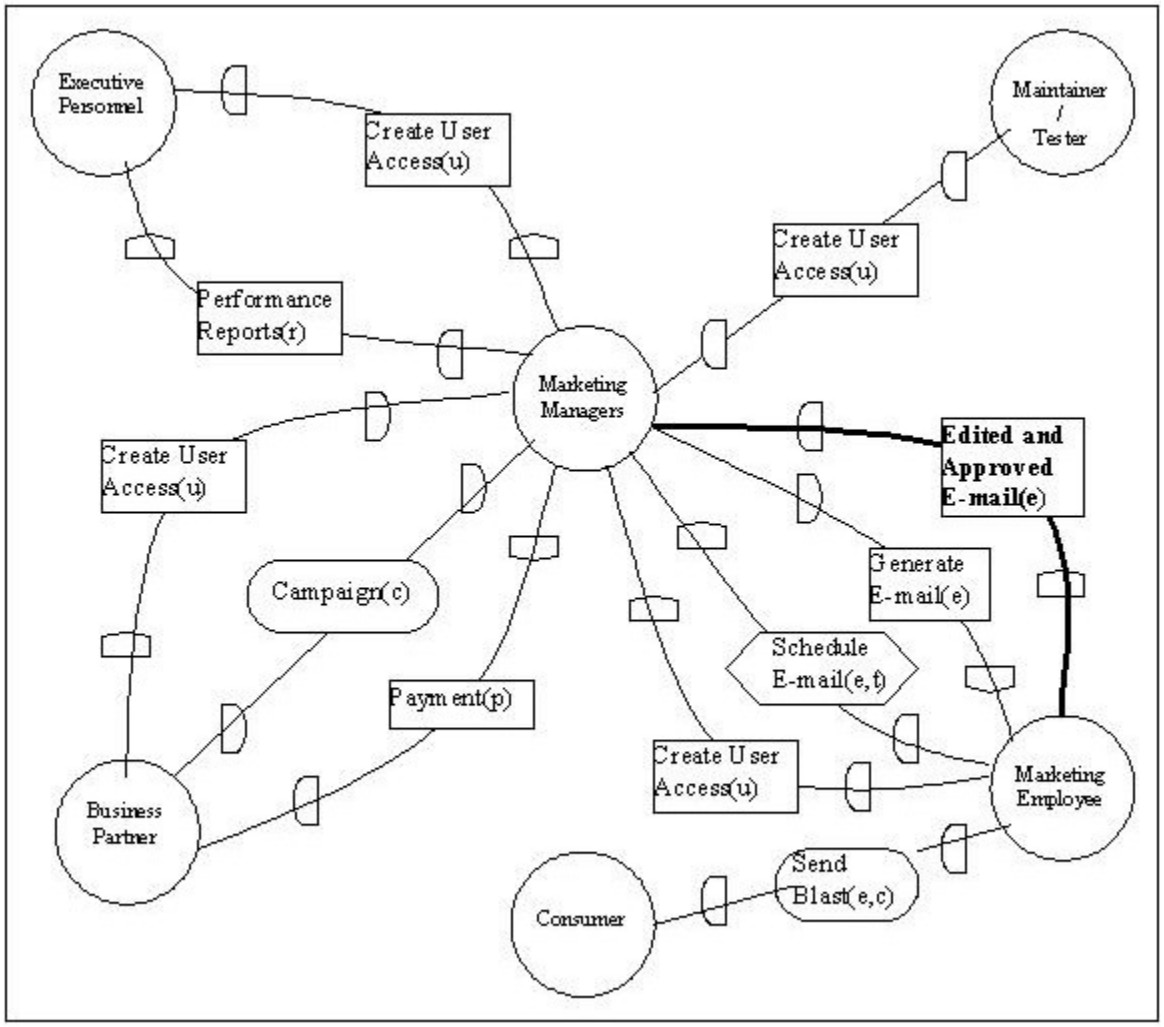

Figure 3.23: Developers' View extracted from the Requirements Negotiations Session, in the Application of the *i** framework

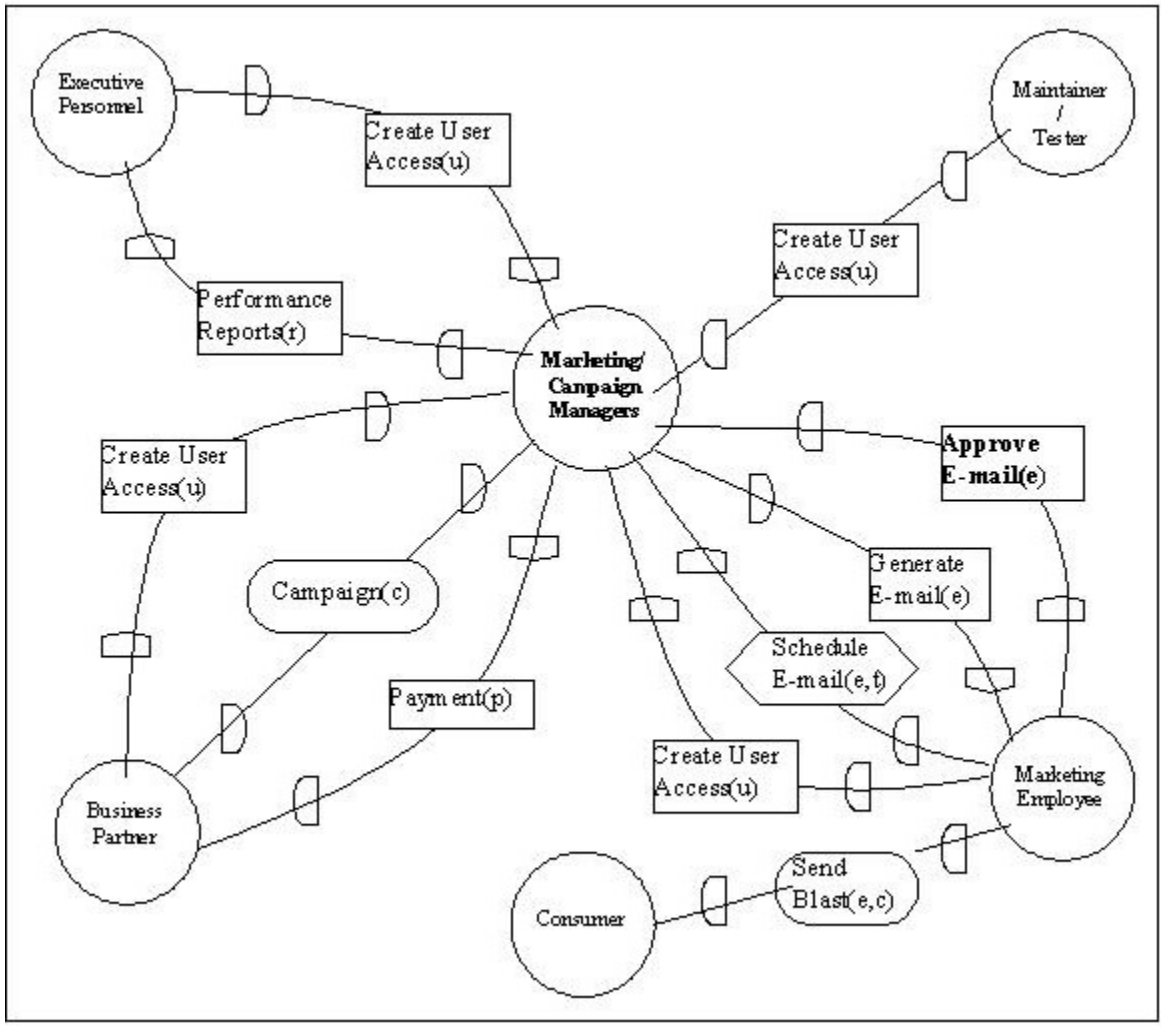

Figure 3.24: Clients' View extracted from the Prototype Demonstration Session, in the Application of the ***i**** framework

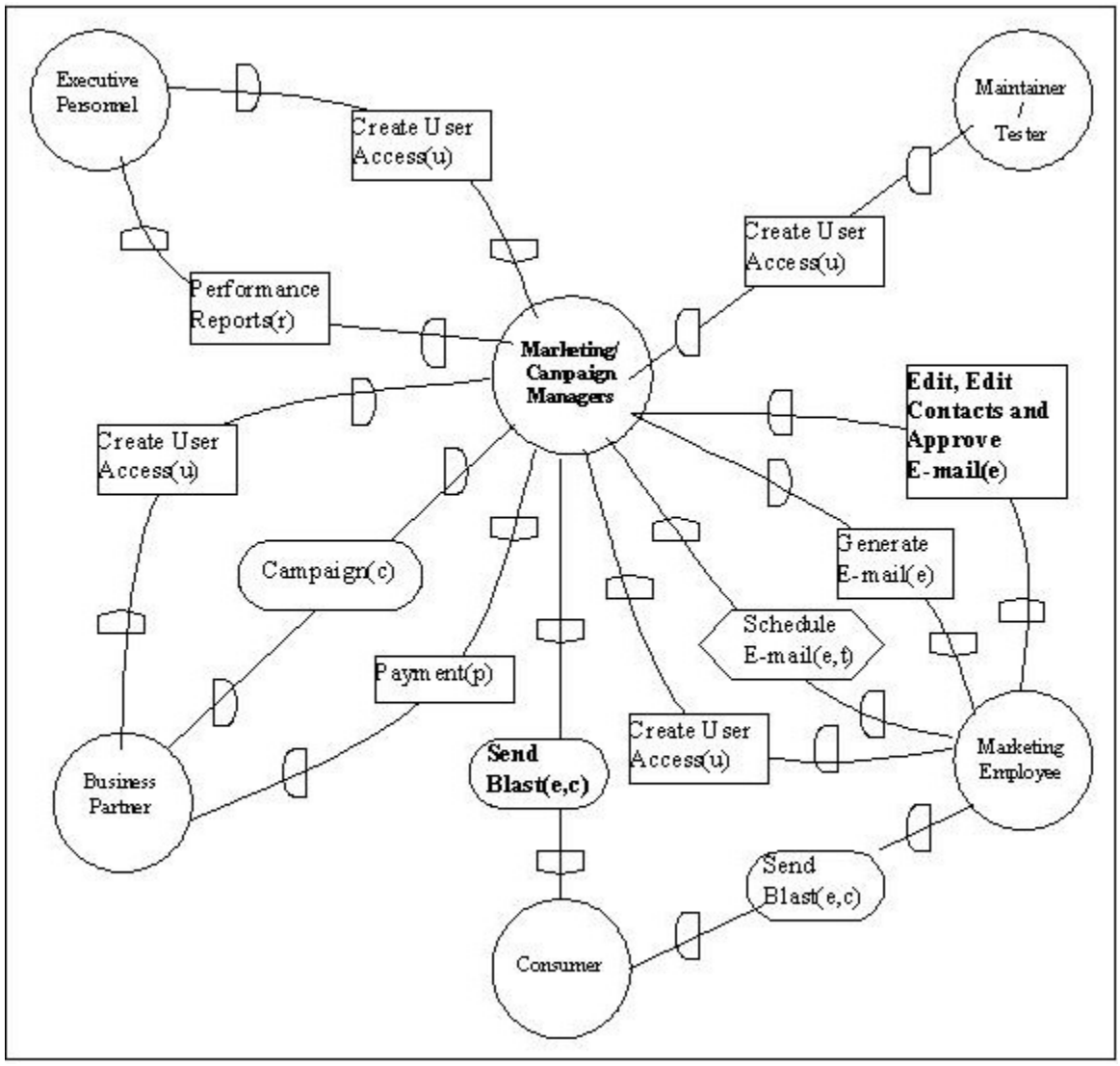

Figure 3.25: Developers' View extracted from the Prototype Demonstration Session, in the Application of the ***i**** framework

prototype was being unveiled, the client had several questions about the functionality of the software and the role of different users in the system.

In the prototype the Developers showed that the marketing manager is able to generate an e-mail to be sent. The Developer believed Marketing Manager has all the capabilities of an employee; likewise an executive should have all the capabilities of a manager. The Client did not like these cascading roles. The Client specified that they want only employees generating e-mails. Even though the manager can schedule when an e-mail should be generated, he or she must delegate the responsibility of actually creating the e-mail to an employee or another manager.

The Developer also gave the manager power to change the addresses the e-mail blasts are sent to. The Client also made clear that they want a manager to do only approval. Approval of the content of the e-mail and approval of the contact lists to be sent but not any changes. A feature was added in which a manager, if he or she wants, to is able to look at the different contact lists that are being used.

In Figure 3.25, the developer wants the managers to have all the functionality of an employee. The view of the client did not change from the negotiation session to the prototype demonstration; they want the marketing employees' functionality to be separate from the manager.

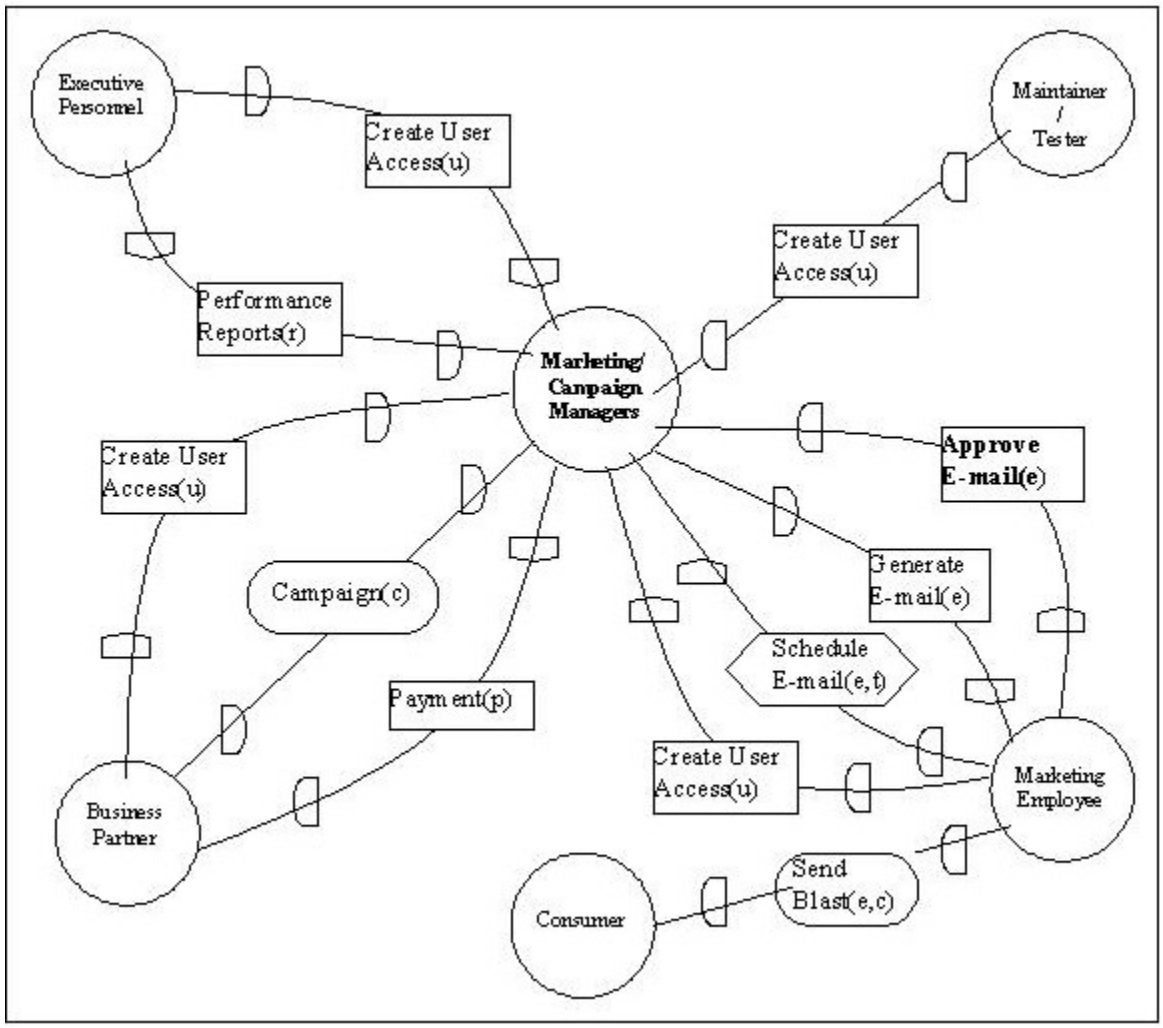

Figure 3.26: Final Resolved View, in the Application of the *i** framework

Requirements Specification version 2.0

Figure 3.26 is the final visualization of the negotiation process. This visualization is a similar to the Client's view in the prototype demonstration of Figure 3.24.

Observations

Throughout the negotiation process, a certain conflict kept coming up, which was the role of users who log on to the system. This conflict arose through the fact that the client wanted the system to be able to create different levels of user access by system administrators. The Developer and Client had differing views on what the functions of and relationships among the different users were. The following are some examples of conflicts that were modeled: (1) A major conflict that lingered right to the end of the process was the role of a Manager. The Client wanted the Managers to be only approving e-mails for distribution and not editing or creating, while the Developer believed right until the prototype demonstration that the Manager should have editing functions. At the end the Developer accepted all the Client's demands with no compromises. (2) The next conflict was that during the elicitation session whether to give the Systems Maintainer full access as envisioned by the Developer or restricted access as envisioned by the Client. (3) Another question was who initiates advertising campaigns. The Client believe that all advertising campaigns should be started by the Manager, while the Developer believed it should be the employees. In

the end the Clients' view prevailed without exception for all the problems described.

3.6 Summary

This chapter's primary function was to describe the case study in great detail. The first section described the context in which the case study was performed. The second section is a description of the experimental data to which the models were applied. In subsequent sections, the application of the models was detailed. For each model, the procedure used in the application, explanation of the input data and the output data included in charts and graphs are presented. The results of the case study are interpreted and the implication of these results are discussed in Chapter 4.

Chapter 4

Discussion

This chapter discusses the insights gained from examining the three models. These insights come from two sources: the literature review and the case study. Chapter 2 and Chapter 3 cover both of these sources respectively. This chapter's primary goal is to interpret the results and to propose a hybrid model based on the individual models working in tandem. The next three sections discuss each model individually. In each section, the results of the model application are interpreted and insights with respect to the usefulness of each model to characterize the conflicts are discussed. Then the model-applied results are compared to the model-free results of the case study. This is followed by a discussion of limitations of each model. The limitations can be categorized into three groups: inherent flaws of the method; constraints of the case study itself; and observer bias. Section 4.4 discusses the hybrid model constructed

from components of the three individual models. The models need to work in tandem because they all have limitations in certain areas and they can complement each other. A hybrid model that takes into account the limitations of each model.

At the beginning of the case study three methodology questions were posed for each model to answer. The three questions were:

1. Is the model useful in characterizing the conflicts?

2. Is the model able to visualize the conflicts?

3. Is the the model able to generate solutions for conflict resolution?

The next three sections will provide answers in the context of these three methodological questions.

4.1 The Utility Curves Model

In terms of the questions that were posed in the case study, the model is unable to characterize or visualize the conflicts of the case study, but it is able to find resolutions to potential conflicts. From the literature[10], it is apparent that the Utility Curves Model inherently focuses on finding a resolution without addressing the conflicts of the stakeholder groups. It is not surprising that the Utility Curves Model cannot characterize, identify or visualize conflicts because it inherently avoids

conflicts completely and focuses on resolution. The model is able to solve requirement conflicts by proposing optimal requirements set before any conflicts are detected.

The raw results of the application of the Utility Curves Model can be found from Figure 3.4 to Figure 3.13, when each figure represents a requirement category. To recount the requirement categories were functional, consumer database, manager approval, performance measures and user-interface requirements. Table 4.1 tabulates the results of each category when the requirements are weighted and Table 4.2 displays the results when weights are not taken into account. In the tables, the combination number chosen by the model-free negotiators, and the calculated using the highest utility and equal utility criteria are presented. For example on the fifth row of Table 4.1, the category is user interface, and the optimal combination determined by the model-free negotiators was C1; while using the Utility Curves Model the combination producing the highest utility was C2 and producing equal utility was C2.

For further investigations, we decided to do some comparisons. First we compared the combination with the highest utility with the combination having equal utility for each category. The second comparison made was between the results when weight was considered and the results when weight was not an input. The third comparison is between the model-free results and the calculated results.

Table 4.1: Weighted Utility Curve Combination Chosen

Requirement Category	**Model-Free**	**Highest Utility**	**Equal Utility**
Functional	C2	C2	C2
Consumer Database	C2	C2	C4
Manager Approval	C3	C3	C1
User Interface	C1	C5	C7
Performance Measures	C8	C8	C7
Agrees w/ Model-Free	–	80%	20%

Table 4.2: Un-Weighted Utility Curve Combination Chosen

Requirement Category	**Model-Free**	**Highest Utility**	**Equal Utility**
Functional	C2	C2	C2
Consumer Database	C2	C2	C4
Manager Approval	C3	C3	C4
User Interface	C1	C5	C7
Performance Measures	C8	C8	C6
Agrees w/ Model-Free	–	80%	20%

4.1.1 Highest Utility Versus Equal Utility Results

In order to help the stakeholders determine the combination of requirements to be included in a Requirements Specification document, two solutions were calculated based on two criteria: the combination with the high total utility and the combination that equally satisfies both stakeholder groups. Which criteria to use should be determined by the negotiating parties beforehand, based on their values.

The combinations calculated using both criteria were compared. From Table 4.1 one can infer by using the Weighted Utility Curves model, only in one category out of five the combination chosen with both criteria was the same. In the Unweighted Utility Curves model, again only in one category out of five the combination calculated with both criteria was the same; refer to Table 4.2. Theese results show that the same category is chosen only 20% of the time.

4.1.2 Weighted Versus Unweighted Results

The difference between Weighted Utility Curves and Unweighted Utility Curves was also analyzed. Weights were originally introduced in order to recognize that not all requirements are equally valuable. Combinations produced by the highest utility criterion for both the weighted and the unweighted was the same across all categories. For combinations produced by the equal utility criterion, two of the categories had different calculated utilities between the weighted and the unweighted. Three out of

five categories or 60% had the same combination.

More studies need to be done in order to determine whether having weights is valuable in the Utility Curves Model, because there is no difference in weighted and unweighted results for the highest utility criterion while there is a 40% difference for the equal utility criterion.

4.1.3 Comparison to Model-Free Results

The combination of requirements chosen by the method-free negotiators was compared with both combinations of requirements calculated by the model. For each category the model decided on a combination that resulted in the highest overall utility and a combination that resulted in both stakeholder groups being equally satisfied.

Table 4.1 suggests that the model, using the highest utility criterion, can determine the same combination as the model-free negotiations 80% of the time. Another way to state this is that instead of spending a great amount of time, money and effort in negotiations to find a good combination of requirements to put in a SRS, an analyst could find a good combination by calculating the combination with the highest utility and have the same results as a model-free negotiations 80% of the time. The rate in determining the model-free combination using the equal utility criterion is 20%.

Table 4.2 compares the the Unweighted Utility Curves results to the model-free

results. Essentially, the combinations chosen by the unweighted model remains almost similar to the categories chosen by the weighted model. The overall result shows that the unweighted model's success rate in predicting the model-free combination is essentially the same at 80% using the highest utility criterion. The rate in determining the model-free combination using the equal utility criterion, also, remains unchanged at 20%.

One important insight gained is that having all stakeholder groups have equal utility is not a good indicator of the model-free combination. As one can see in both Table 4.1 and Table 4.2, using the equal utility criterion determines the same combination as the model-free negotiations only 20% of the time, while the highest utility criterion determines the same combination as the model-free negotiations 80% of the time. Therefore, the highest utility is probably the better criteria used for the Utility Curves Model.

4.1.4 Limitations in Applying the Utility Curves Model

The following are the limitations in applying the Utility Curves Model. The limitations of this model are separated into three categories: inherent limitations of the model, constraints of the case study, and observer limitations and bias.

Inherent Flaws

L.UC.1 The Utility Curves Model inherently focuses on finding a solution and avoids identifying conflict, which according to Easterbrook[1] leads to potential disaster in the later stages of software development. The model emphasizes that a solution be found immediately before conflicts can be found in communication session. This model by design can neither identify nor characterize requirements conflicts. This inability to acknowledge conflicts is inherent in this model because the designers and wanted to focus on finding the the "optimal" resolution.

L.UC.2 Although the Utility Curves Model can be graphed in Cartesian coordinates, this model cannot visualize requirements conflicts. In order to graph utility curves there must be a limit of two stakeholders, because each stakeholder represents one dimension and there are two dimensions in a Cartesian graph. It proves difficult to visualize in three or more dimensions. When graphed in two dimensions, the utility curves are not intuitive to understand and does not provide additional aid in solving the problem. This is the reason utility curves were not graphed in the case study.

L.UC.3 The inability of the model to fully capture the dynamic nature of requirements is a major limitation. The model assumes that the level of utility and weight for each requirement to remain static. In other words, the amount a stakeholder

group values a requirement cannot change over the requirements analysis process. Also it does not take into account the fact that new requirements are added and old requirements are removed. Robinson wrote that "Utility theory is only relevant when people maintain consistency in their preference of alternatives;"[26] This model can capture how much requirements are valued at any given time, but not over a length of time.

L.UC.4 Initial requirements list needs to be very explicit in order for stakeholder groups and analysts to assign utility values and weights. Each item on the list of requirements should be a single atomized requirement. One item in the requirements list may be stated as one requirement but there may be two or more requirements hidden in the wording. Other problems for requirements elicitation include not writing very simple requirements that may seem obvious or the very vague requirements. This model is unable to deal with implicit requirements that may be hidden in the text of a requirement, not stated at all or written very vaguely. Writing explicit requirements is not easy to do for the stakeholder groups or the analyst with the lack of elicitation tools. In future research requirements elicitation methods should be evaluated. In the case study the stakeholder groups may have tried to explicitly state the requirement, but some requirements were inferred from the wording or were dropped because they were not really requirements.

Case Study Constraints

L.UC.5 The case study is constrained to a two-stakeholder software project. In the real-world there could be many more stakeholder groups other than the developer and the customer/user. Others include external businesses that could benefit from the success of the Media Distribution Platform, software maintainers who will repair an update the product from time to time, and business executives who are employers of the developer and customer. Adding these stakeholder groups would have added a few more dimensions to the problem, but would be more accurate of real life.

L.UC.6 There were about 30 explicit requirements that were modeled overall. In a real world application there may be hundreds of requirements. So there is an issue of how well this model will characterize and solve conflicts when there are hundreds of requirements and several stakeholder groups.

L.UC.7 The case was studied five to six months after actually taking place. This model is more practical if the assignment of input values were done during the negotiation process. Also the assignment of values should have been done in consultation with the stakeholder teams. The stakeholder groups would have better knowledge of the utility and weight of each requirement.

Observer Bias and Limitations

L.UC.8 All the input values, utility and weight, were assigned by a single external researcher after the actual negotiations took place. The fact that the researcher was external to the project being studied was a strength because the researcher was not a member of either negotiation team and does not favor one team over another. It should be noted the investigator personally knew two members of the Developer team. Having one researcher is a major constraint because the researchers personal biases and views may skew the input values. A researcher may have different ideas about trends in software engineering. Having more than one researcher could have balanced out some of these biases.

L.UC.9 Another difficulty in implementing this model is that it is hard to capture quantitatively how much a stakeholder group values a certain requirement. In the case study, there was extensive videos and documentation of the negotiation sessions, but it was difficult to place relative value on many of the attributes.

4.1.5 Summary

The Utility Curves Model is capable of finding solutions to requirements conflicts without being able to characterize these conflicts. This model does not seek to find which models are in conflict, but rather place all the requirements in the model and

let the model determine the optimal requirements set. Table 4.3 summarizes the capabilities and limitations of the Utility Curves Model. Because model does not characterize conflicts at all, the model is also unable to identify or visualize conflicts. The model does not respond to changes to the requirements because solutions are found immediately without going through a process. Although the model does not require expert analysts or stakeholder collaboration, they are still useful. Additions will be made to the Table 4.3 after every subsequent model.

4.2 The Win-Win Model

The first methodology question was if the conflicts can be characterized by the model. Yes, the model does characterize each conflict as a single unit. The model unfortunately is unable to identify conflicts or to list of viable resolution options. In order to identify the conflicts a human operator, usually the stakeholder groups or an agent representing them, is required. The listing of viable solutions also needs to be done by human operators, which in the case study was performed by the stakeholder groups negotiating with each other. The model is unable to visualize the conflict, to answer the second methodological question.

The third methodological question was whether conflicts can be solved. The Win-Win Model, using the Cost-Benefit Analysis Method (CBAM), is able to solve conflicts, and the case study can support this proposition. In the first iteration,

Table 4.3: Capabilities of the Models

Capability/ Limitation	**Utility Curves Model**
Automatically Identify Conflicts	no
Characterize Conflicts	no
Visualize Conflicts	no
Address Requirement Change	no
Need Expert Analyst	no
Need Stakeholder Collaboration	no
Find Solutions	yes

there were 21 requirements conflicts that needed to be resolved. The model was able to find a solution for all 21 conflicts. 8 of the conflicts had solutions that were clear "winners" and 13 of the solutions were only "marginal" winners. In the second iteration, 8 conflicts needed to be resolved and all eight were indeed resolved. Two of the solutions were clear "winners" and six solutions were only "marginal" winners. In the final iteration two conflicts were out for discussion and both these conflicts were resolved. The model was able to find the best solution for both conflicts, but it was solved clearly for one issue but only marginally for the other. Once CBAM finds the best solution, this help to the stakeholder groups more knowledge before adopting a resolution.

The Win-Win Model is able to characterize requirements and conflicts as they change throughout the requirements negotiation process. Because the model goes through several iterations, conflicts can be addressed again if there are any changes, and if new conflicts and requirements arise, they can also be addressed. The Utility Curves Model, in comparison, is unable to model the dynamic nature of requirement conflicts.

4.2.1 Comparison to Model-Free Results

In the first iteration, there were 21 requirements conflicts that needed to be resolved. The model was able to evaluate all the options and suggested the best solution for all

21 conflicts. In the model-free version of these negotiations, 13 conflicts were resolved and 8 conflicts were not. This first analysis supports that having a model guarantees a greater success in at least suggesting the best solution.

The solutions of the 13 conflicts that the stakeholder groups resolved in the model-free negotiations will be compared with the solutions of the model. For 9 of the 13 conflicts the Win-Win Model suggested the same solution as the model-free negotiators. For 4 of the 13 conflicts the Win-Win Model found different solutions than the model-free negotiators with marginally certainty.

We will discuss why the model-based solutions to four conflicts were different from the model-free solutions. For conflict 1.14 from Figure 3.15 the model-free negotiators chose option A, which is costlier, while the model chose option B. The difference can be attributed to the fact that one option was prohibitively costlier than another. The model-free negotiators did not take the cost into account when they chose option A, while the model chose the cheaper option. Using the model probably forces the negotiators think about the costs because they must enter a cost value for every option. For conflict 1.2 from Figure 3.14, and 1.18 and 1.20 from Figure 3.15 the model-free solutions was found by the negotiators choosing the option favored by the client, because it was the easiest way to suppress these conflicts. The model chose the options favored by the developer because the calculations indicate these options had greater desirability.

In the second iteration, the eight conflicts not resolved in the initial model-free iteration were re-open for discussion. The method was able to again find the best solution for every single conflict. In the model-free version of this iteration, six out of eight issues were resolved and two out of eight were unresolved. For Five out of the six conflicts the Win-Win Model found the same solution as the model-free negotiators. Only conflict 2.4 from Figure 3.16 had a solution different from model-free. This difference can be again attributed to the fact that the models-free negotiators did not consider cost when evaluating and choosing the options.

In the third and final iteration, two conflicts needed to be solved. Both of these conflicts were issues that were solved in the first iteration, but needed to be reopened for discussion. In the model-free negotiations both issues were resolved. For conflict 3.1 from Figure 3.17 both the model-based and the model-free chose the same solution while for conflict 3.2 both methods came to a different solution. The reason for this difference in conflict 2.2 was that the model-free negotiators went with the option favoring the client to speed up negotiations rather than evaluate the merits of each option.

The following are three benefits of utilizing this Win-Win Model that have arisen in this analysis. The first benefit is that the Win-Win model finds a solution when the stakeholders cannot agree on a solution. We have demonstrated that the model is able to find a solution 100% of the time.

Secondly using the model will reduce the number of situations where a resolution option is picked without evaluation because the negotiators were in a hurry or wanted to suppress the conflict. For four of the conflicts analyzed, model-free negotiators chose the resolution most favored by the client while the model chose a different resolution.

The third benefit is that the model forces negotiators to think about cost. Since the cost of each resolution option must be quantified into procedures of The Win-Win Model, the stakeholders will consider cost implications of each option presented. For two of the conflicts, the model-free negotiators chose the option with a considerably higher cost while the model shows the lower costing option.

4.2.2 Limitations in Applying the Win-Win Model

The following are the limitations in applying the Win-Win Model with Cost Benefit Analysis Method. These limitations are subdivided into inherent flaws, case study constraints and observer bias.

Inherent Flaws

L.WW.1 The Win-Win Model is unable to identify requirements conflicts automatically. These conflicts need to be identified by either a human operator, usually stakeholders or an analyst, or a heuristic model such as the Quality Attribute Risk

and Conflict Consultant (QARCC)[31]. The Win-Win Model a set of procedures used to handle and resolve conflicts. In the case study each stakeholder groups identified their own conflicts. Conflicts are identified in each iteration of the model and the conflicts are characterize by placing them a numbered list.

L.WW.2 The Win-Win Model is unable to explicitly list viable resolution options for each conflict. This list of options must be created by the stakeholder groups in a collaborative session. The list of resolution options for each conflict need to be explicitly written in order for stakeholder groups and analysts to input values. In this case study, the resolution options were explicit for the most part. There were times where a solution was implied but not explicitly stated in the data compilation.

L.WW.3 The Win-Win Model by itself is unable to determine the best solution among the list of viable solutions. A set mathematical methods and/or a series of negotiation sessions is required to find the optimal solution. In the case study, Cost-Benefit Analysis Method, a mathematical tool, was used employed to calculate the solution with the greatest desirability.

L.WW.4 The Win-Win Model has no mechanism to visualize the conflicts it characterizes. Another model that can visualize the conflicts should be used as a complement to the Win-Win Model. Having visualizations could be a benefit

to conflict resolution, because when a human operator can visualize the conflicts it makes it easier for the stakeholder groups to identify conflicts and list of viable resolutions. The stakeholder groups may be able to identify more conflicts and resolutions using visualizations than when the requirements are written in words. The ***i**** framework is a visualization model that can be used with the Win-Win model.

L.WW.5 The Win-Win model fails to recognize the interconnectedness of requirements and their conflicts. This model solves each conflict individually to the exclusion of other conflicts. The resolution of one conflict may have an effect on another conflict but the model will not be able to recognize such an effect. For example, conflict 1.2 and 1.4 from Figure 3.14 contradicted each other and would not have been caught by the model. In other words the model does not see the system as a whole but rather as individual mutually exclusive components.

L.WW.6 The Win-Win model is heavily dependent stakeholder collaboration. The more stakeholder groups are able to communicate their needs and desires with each other the more successful the requirements process will be. It costs time, effort and money for all the stakeholder teams to be present at all negotiation sessions. If the model was able to decrease stakeholder collaboration that would've been a benefit to the stakeholders because the cost of collaboration would be reduced.

Unfortunately, the model could not decrease stakeholder collaboration.

Case Study Constraints

L.WW.7 The case study is constrained to two stakeholder groups. In this particular case study there could be many more stakeholder groups other than the developer and the customer/user. Others include external businesses that could benefit from the success of the Media Distribution Platform, software maintainers who will repair an update the product from time to time, and business executives who are employers of the developer and customer. Adding these stakeholder groups would have added a few more dimensions to the problem, but would be more accurate of real life.

L.WW.8 The case was studied five to six months after actually taking place. This model is more practical if the assignment of input values were done during the negotiation process, because the model depends on collaboration from all stakeholder groups and analysts. Also the assignment of values should have been done in consultation with the stakeholder teams. It is difficult to say whether it makes any difference when and by whom the values were assigned.

Observer Bias and Limitations

L.WW.9 All the input values, including QAScores, contribution values and costs, were assigned by a single external researcher after the actual negotiations took place. Since the Win-Win Model is a heavily collaborative process, the fact that the investigator in this study was external is a limitation. Having one researcher is a major constraint because the researchers personal biases and views may skew the input values. A researcher may have different ideas about trends in software engineering. Having more than one researcher could have balanced out some of these biases.

L.WW.10 Another difficulty in implementing this model was that it is hard to capture quantitatively how much a stakeholder group values a certain requirement. This was especially difficult because the data compilation was qualitative and the investigator was then able to collaborate with the stakeholders.

4.2.3 Summary

Table 4.4 is a brief summary of the strengths and limitations of the models analyzed so for. The model does not identify conflicts, but the stakeholders themselves do. Once the conflicts have been identified they are characterized as text, but not as visualizations. The model is able to deal with requirements changes because the model

goes through iterations. The Win-Win Model requires strong stakeholder collaboration to work correctly, therefore this is a limitation if all members of stakeholder teams cannot be present. The model is capable of finding solutions to requirements conflicts.

4.3 The *i** Framework

The first two methodology question on whether the ***i**** Framework is useful in characterizing and visualizing conflicts were answered in the positive, the conflicts were characterized through the visualizations. We believe that the visualizations created in the case study were representative of the conflicts in the requirements negotiation process. We created two competing visualizations, one visualization per stakeholder, for every stage of the requirements engineering process and the inconsistency between the two visualizations represented the conflict.

The third methodology question was whether the ***i**** Framework can solve conflicts and was answered in the negative, because we had no method to reconcile conflicting visualizations. This is no surprise because there was no set procedures to resolve inconsistencies between visualizations in previous literature. In the Kids Help Phone case study [35] they used ad hoc methods to resolve inconsistencies and overall were unable to reconcile any visualization.

We were not able to characterize all the conflicts in the case study; we were

Table 4.4: Capabilities of the Models

Capability/ Limitation	**Utility Curves Model**	**Win-Win Model**
Automatically Identify Conflicts	no	no
Characterize Conflicts	no	yes
Visualize Conflicts	no	no
Address Requirement Change	no	yes
Need Expert Analyst	no	no
Need Stakeholder Collaboration	no	yes
Find Solutions	yes	yes

limited to conflicts associated with system actors. An actor-based conflict is when the function of a system end-user is questioned by one or more of the stakeholders. Examples of actor-based conflict include issues like the security access level of a manager or how much of the system can an external developer have. The Strategic Dependency (SD) [34] model that the ***i**** Framework operates under is the reason only actor-based conflicts were characterized. The Framework has the potential to visualize more types of conflict, other than actor-based ones, if ***i**** operated under the Strategic Rationale (SR) rather than SD. More research needs to be done into SR to see if it can cover more conflicts and perhaps solve conflicts.

4.3.1 Qualitative Observations

The following are some qualitative observations that that were made during the application of ***i**** Framework.

It is easier to conceptualize conflicts when they are represented as the visualizations thanwhen they are represented as text or transcripts. The visualizations made it easier for someone to spot the conflict because one can visually see discrepancies when two conflicting visualization are placed side-by-side. If there are one or two discrepancies, we made the areas of conflict with bold lettering and thicker lines made identifying corresponding discrepancy between visualizations quicker. For example place Figure 3.22 to Figure 3.23 side-by-side to observe how the area of conflict was

highlighted with bold lettering.

We believe that if there are more than two discrepancies, a color coordination scheme can be created. For example, when the color red is used in the two conflicting visualizations one knows that the areas colored in red correspond with each other. The color scheme will enhance the visualization of conflicts.

Another interesting observation was that the ***i**** Framework is able to handle requirements changes throughout the planning stages. Requirements change occur because a stakeholder initiates a change or the stakeholders through their negotiations change a requirement. When a requirements changes the change must be reflected in the visualization. In the case study we demonstrated that by using the ***i**** Framework, changing a visualization to correspond with requirements changes requires little activity.

The most common change was when the relationship or dependency between any two actors is changed. Such a change requires minimal activity because the only change in the visualization is adding and removing lines, or changing the direction of the arrows. For example between Figure 3.19 to Figure 3.5.2 relationship lines between actors are removed.

The second most common change was when actors are added or removed must change more activity is required to change the visualization. Simply adding and removing an actor in the visualization will require more activity than a relationship

change. When an actor is added or remove relationships would also have to be added to remove, hence the extra activity. The addition and removal of actors was demonstrated between Figure 3.18 to Figure 3.19.

Sometimes several requirements changes occur resulting in the major restructuring of the visualizations. Only one major restructuring of the visualization from Figure 3.5.2 to Figure 3.21 occurred in the case study. In this restructuring we had to re-create the visualization from scratch rather than changing an older visualization.

Altering the visualizations is simple if it is a relationship change, a little more complex when a single actor is removed or added. When there are several requirements changes then the visualization must be re-created rather than altered.

To recap, through the application of the ***i**** framework, we were able to determine that visualization of actor-based conflicts is possible, but cannot solve such conflicts. The framework is able to address changes throughout the requirements engineering phase because it is easy to change the visualizations that represent the system.

4.3.2 Comparison to Model-Free Results

Since the ***i**** framework could not find solutions to the conflict it was modeling, results could not be produced to compare with the results of the model-free negotiations.

4.3.3 Limitations in Applying the *i** Framework

The following are the limitations in applying the the ***i**** Framework. These limitations are subdivided into inherent flaws, case study constraints and observer bias.

Inherent Flaws

L.IF.1 The ***i**** framework is unable to resolve requirements conflicts as a stand-alone model. Since the framework can visualize conflicts, this model can be used in tandem with a model that can solve conflicts well.

L.IF.2 Whoever uses the ***i**** framework should be well versed in using the framework. The model has many fine details when producing the visualizations and requires an expert analyst. It is not simple for anyone to learn the graphical or the formal representation of ***i****. An analyst who is knowledgeable in ***i**** needs to be found.

L.IF.3 Only visualization of actor-related conflicts were modeled. The Strategic Dependency (SD) model of ***i**** framework could not model conflicts that were not actor-related. The framework claims that all the system requirements can be modeled using the Strategic Rationale (SR) model.

Case Study Constraints

L.IF.4 The case study is constrained to a two-stakeholder software project. In this particular case study there could be many more stakeholder groups other than the developer and the customer/user. Others include external businesses that could benefit from the success of the Media Distribution Platform, software maintainers who will repair an update the product from time to time, and business executives who are employers of the developer and customer. Adding these stakeholder groups would have added a few more dimensions to the problem, but would be more accurate of real life.

L.IF.5 There were at most eight types users that were modeled in the ***i**** visualizations. In the real world there may be dozens of users of an application. It may not be possible to visualize more than 20 different types of users, because the visualizations will become too crowded or may have to be broken up into several images. Also, the analysis of a visualization will prove cumbersome as it did in the original ***i**** Framework case study[35].

Observer Bias and Limitations

L.IF.6 All the visualizations were produced by a single external researcher after the actual negotiations took place. The fact that the researcher was external to the

project reduced any bias the researcher has towards either stakeholder team. Having one researcher is a major constraint because the researchers personal biases and views may skew the input values. A researcher may have different ideas about trends in software engineering. Having more than one researcher could have balanced out some of these biases.

L.IF.7 The investigator in the case steady had rudimentary knowledge of the ***i**** Framework. as mentioned earlier, someone using the framework should have extensive knowledge. The investigator may have not been able to exploit this model to the full extent as it has many nuances.

4.3.4 Summary

In the ***i*** framework requirements conflicts are identified with the help of stakeholders and are characterized through visualizations made by an expert analyst. Unfortunately the model is unable to solve conflicts on its own. The main benefit of using the framework is that requirements changes can be addressed because altering the visualizations that represent the system is not difficult. An expert analyst is needed to create the visualizations. Stakeholder collaboration is not required but will help the analyst.

The strengths and limitations of the all three models is summarized in Table 4.5. The Capabilities/Limitations listed in the left column of Table 4.5 are the main

Table 4.5: Capabilities of the Models

Capability/ Limitation	**Utility Curves Model**	**Win-Win Model**	***i**** **Framework**
Automatically Identify Conflicts	no	no	no
Characterize Conflicts	no	yes	yes
Visualize Conflicts	no	no	yes
Address Requirement Change	no	yes	yes
Need Expert Analyst	no	no	yes
Need Stakeholder Collaboration	no	yes	yes
Find Solutions	yes	yes	no

criteria we believe should be included in an ideal model that addresses requirements conflicts. These criteria should be included in a hypothetical hybrid model built out of the three individual models.

The first criterion is the ability to automatically identify conflicts was identified in literature by Robinson[10] as the most important. Unfortunately a hybrid model constructed out of the three models cannot perform this criterion, because the three models are incapable.

The criterion to characterize the conflicts is important because there are many dangers to avoiding conflict according to Easterbrook[1]. An ideal model should at least represent each conflict.

The ability to visualize conflict was one of the methodology queries and we believe should be an integral part of an ideal hybrid model.

The importance of a model to work while requirements change became apparent when applying the models in the case study. The Win-Win Model and the ***i**** Framework were able to address requirements change and will be leveraged in the hybrid model.

The needs for an expert analyst and stakeholder collaboration are limitations for any model, because either need is not readily available. A hybrid model ideally should not have these needs, but will unfortunately inherit them from the individual models.

The ability to find solutions to conflict was one of the methodology queries and is

also important identified by Robinson[10] as important conflict resolution trait. The hybrid model will have this characteristic by leveraging the Utility Curves Model and the Win-Win Model you have the ability to find solutions.

4.4 Models in Tandem

Because a model has limitations, one model working individually is not sufficient for a software planner deal with requirements conflicts. One model alone cannot identify, visualize, characterize and solve requirements conflicts in order to write a good specification document. The need for the most work in tandem is bourne from the limitations of the individual models. See Table 4.5 for a brief summary of what the three models are capable of and limited in performing. The models working in tandem will enhance conflict resolution. Firstly several processes are described on how the three models complement each other. Second the procedures of a possible hybrid model that incorporates Utility Curves model, Win-Win Model and the ***i**** Framework are described.

4.4.1 List of Complementary Proccesses

CP.1 The procedures of the Utility Curves Model should be performed in parallel to the procedures of the Win-Win Model. These two models will complement each

other well: the Utility Curves Model cannot characterize individual conflicts (Limitation L.UC.1 in Section 4.1.4) while the Win-Win model does characterize individual conflicts; and the Win-Win Model fails to recognize the system as a whole, solving conflicts to the exclusion of others, (Limitation L.WW.5 in Section 4.1.4) while the Utility Curves Model does solve the problem as a whole by determining the best combination of requirements that work together. Having the two models work in parallel brings a wholesome perspective and a dispersed perspective to conflict resolution. Also, both models are capable of finding solutions and both working together reinforces the solution. The final requirements set determined by the Utility Curves model should be compared with the final winning requirements calculated by the Win-Win model. If the same requirements appear on both lists, this will reinforce the need to keep this requirement in the specification. When there are discrepancies in the lists, this is both advantageous and disadvantageous. The advantage is that this will allow stakeholder groups and analysts to review possible errors in the requirements process. The disadvantage of these discrepancies are that extra negotiation sessions may need to be convened or a requirements will take more time to re-examine the discrepancies.

CP.2 When a solution to a conflict needs to be found from a list of resolution options in the Win-Win Model, the Utility Curves Model should be used to calculate the

optimal solution along with the Cost Benefit Analysis Method (CBAM). This is required because the Win-Win Model is unable to solve conflicts by itself (Limitation L.WW.3 in Section 4.2.2) and CBAM and the Utility Curves model is unable to characterize individual conflicts (Limitation L.UC.1 in Section 4.1.4). For every conflict in each Win-Win iteration there are two or more resolution options to evaluate. In the case study it was demonstrated that CBAM can be used to find the optimal solution; the Utility Curves model should also be used to evaluate the options using some of the same inputs as CBAM. This complementary process reinforces the results when both CBAM and Utility Curves Model recommend the same solution. When the two models choose different resolution options, this gives stakeholders an opportunity to discuss the discrepancies and eventually adopt the winning resolution through negotiation or reapplying these models to the list of resolution. There is a precedence in literature where weighted average formula, similar to the one used in the Utility Curves Model, is employed in the Win-Win Model at the stage a resolution to a conflict needs to be found. The Multi-Criteria Preference Analysis[22] assesses different options and calculates an optimal solution within the context of the Win-Win Model. The weighted utility curves formula, Equation 2.1, needs to be modified in order to fit the CBAM inputs. Eliciting more inputs from the stakeholders is not desirable because this will add more pressure to the stake-

holders than is already present. The CBAM values can be easily transferred to the Utility Curves Model. The utility value is the contribution value generated for CBAM plus 1 in order to avoid negative numbers; and the the weight is simply the QAScores needed for CBAM. Equation 4.1 is the modified weighted average formula.

$$U(R_i) = \frac{\sum_j ((Contribution_{ij} + 1)(QAscore_j))}{\sum_j QAscore_j} \tag{4.1}$$

CP.3 For every iteration of a conflict in the Win-Win Model, an ***i**** visualization can be created to help negotiators understand the conflict better and list viable solutions, after which Cost-Benefit Analysis Method (CBAM)[33] will choose the optimal solution. Because the Win-Win model does not have a visualization technique (Limitation L.WW.4 in Section 4.2.2) and the ***i**** Framework cannot solve conflicts (Limitation L.IF.1 in Section 4.3.3), adding ***i**** as a complement to Win-Win makes good sense. Only actor-based conflicts can be visualized due to limitations of the ***i****framework (Limitation L.IF.3 in Section 4.3.3). Once an actor-based issue or conflict has been identified by the stakeholder groups in the Win-Win Model, an ***i**** visualization can be drawn to characterize the conflict. Either one of visualization can be drawn by the analysts or competing visualizations can be created for every different stakeholder group. The visualization helps negotiators and analysts easily conceptualize the conflict at hand.

The visualization should also be used as an aid to draft the list of resolution options. The stakeholder groups and the analyst will still have to manually list the resolution options, but are aided by the ***i**** framework. CBAM will be used to determine the best option; if it is not successful another Win-Win iteration must be made to try to resolve the issue again. With ***i****, if alterations need to be made to the visualization and only the dependency changes need to be made, existing visualizations can be easily altered for the next iteration. This is limited to one type of conflict for now; perhaps in the future work if other types of conflicts can be characterized by ***i****.

CP.4 Since the requirements negotiation process will now involve the Utility Curves Model, the Win-Win Model and the ***i**** Framework, it is inevitable that the process requires heavy involvement of all the stakeholder groups and a facilitator who is an expert in all three models. This is because the Win-Win Model requires heavy stakeholder involvement (Limitation L.WW.6 in Section 4.2.2), while the ***i**** Framework requires an expert analyst (Limitation L.IF.2 in Section 4.3.3). Having an facilitator who is an expert in all three models may reduce the burden on the stakeholder groups who need to be present at all negotiation sessions. Likewise, having some stakeholder involvement will slightly reduce the burden on the facilitator. The Utility Curves Model does not require an expert facilitator or heavy stakeholder involvement, but having these factors will make

the model more effective. Increased consultations among stakeholders groups and negotiation facilitators will lead to better characterization of conflicts and better solutions.

4.4.2 Hybrid Model

In this section a description of a hybrid model consisting of components of the three models is given, which is based on the complementary processes detailed in Section 4.4.1. The ultimate goal for the stakeholders is to have a software product that satisfies all the stakeholders' requirements. The first stage to achieve this goal is for the stakeholders to collaboratively draft a Requirements Specification document that states the requirements that are satisfactory to all stakeholders. The purpose of the hybrid model is to create a final requirements set from an initial requirements set that satisfies all stakeholders. The hybrid model will be used to guide the stakeholders through negotiations as they seek to create a conflict-free set of requirements to write into the Requirement Specification document. The model guides the stakeholders through the process of identification and resolution of conflicts and provides tools to address the conflict. This hybrid model should not to be used as a postmortem analysis model as it was done in a case study, but rather a real-time negotiation model. Figure 4.1 is a visual representation of the hybrid model.

The personnel required in the negotiations are representative teams of the stake-

holder groups and an analyst who is well-versed in the three individual models and the hybrid model. CP.1 in Section 4.4.1 explains the reasons for requiring stakeholder and analyst involvement.

The hybrid model has two parallel processes, one process that automatically find the solution without stakeholder collaboration and another process that has stakeholder collaboration. The automatic process uses the procedures of the Utility Curves Model and the collaborative process follows the Win-Win model. The Utility Curves Model is the first process to be deployed to have an immediate solution, which will be compared to the results of the collaborative process. CP.1 in Section 4.4.1 explains the rationale for having these parallel processes.

In the collaborative process, the Win-Win Model is used as the general framework to guide the process. The Win-Win Model uses other methods to address conflicts. The ***i**** Framework is deployed at the conflict characterization step (CP.3 in Section 4.4.1). The Cost-Benefit Analysis Method and Utility Curves is employed at the conflict resolution step (CP.2 in Section 4.4.1).

Step 1 Initially the requirements of the system reside as ideas with the stakeholders. In this step each stakeholder team converts these ideas into an explicit list of requirements. Each requirement should be written as explicit as possible. Writing explicit requirements is not a simple task; see Limitation L.UC.4 in Section 4.1.4 about the difficulties in writing explicit requirements. Each stakeholder team

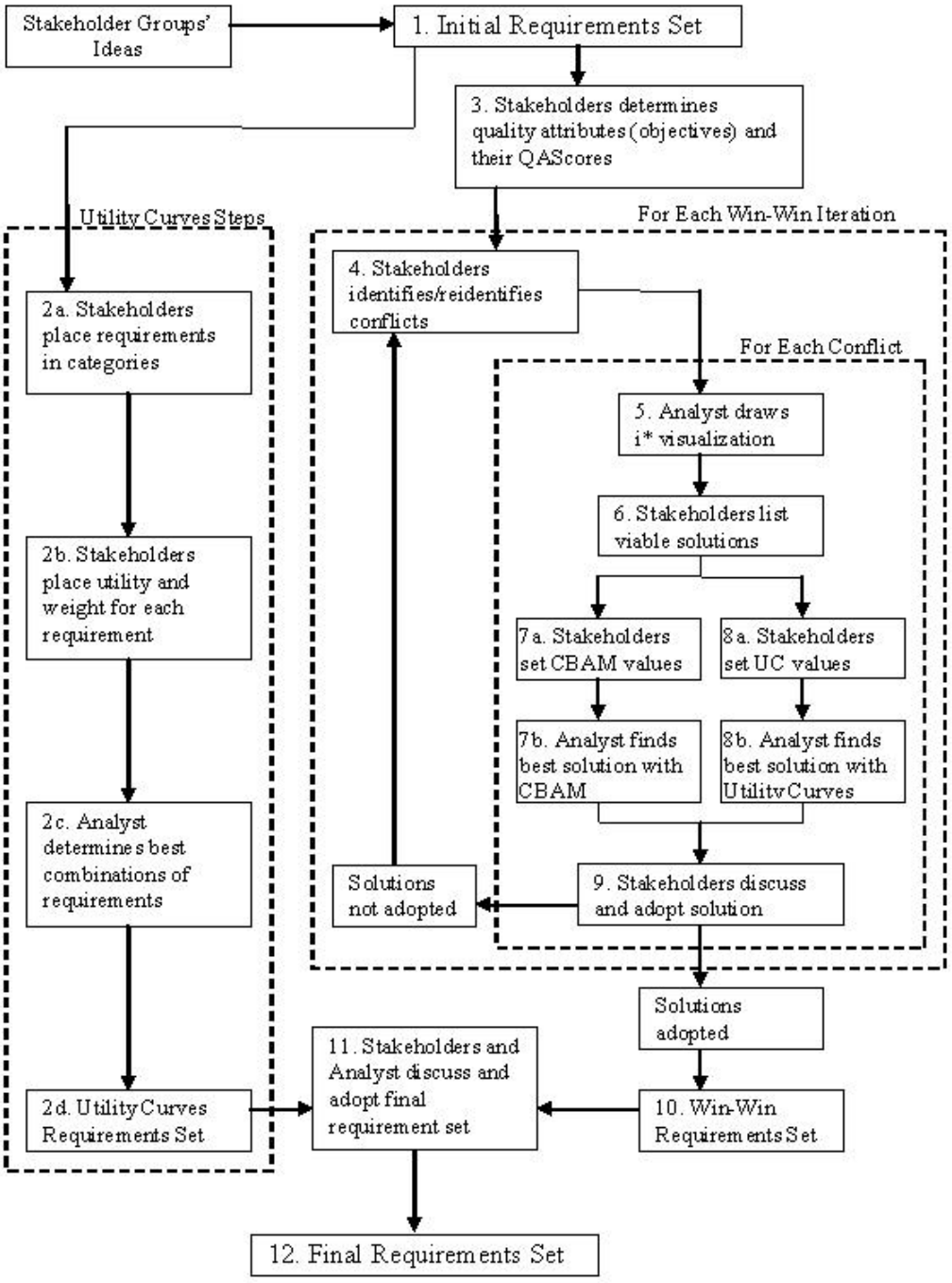

Figure 4.1: Hybrid Requirents Conflict Resolution Model

may have to meet a few times before releasing their initial list of requirements.

Step 2 In this step the procedures of the Utility Curves Model is employed in order to find a requirements set automatically. See Section 2.3.1 for more information on the procedures of the Utility Curves Model.

a In this step all the requirements from all the stakeholders are divided into categories. This is done to break up the problem and make it more manageable. Dividing the requirements into categories may not always be possible if the requirements from separate categories interact with each other. The requirements used are all the requirements from all stakeholders. This procedure may be performed by the analyst or the stakeholders meeting in a session.

b For every explicit requirement in every category each stakeholder group should assign a utility and a relative weight. Utility is a value between 0 and 1, where 1 is the highest utility. Relative weight is based on the importance of the requirement and is a scale between 1 and 10, 10 being of highest importance. This task should be performed by the stakeholder teams themselves.

c The analyst should initially make several plausible combinations of requirements in each category and then calculate the best plausible combination

using the weighted utility curves formula, see Equation 3.1. Where U^P is the utility of combination, P, U_a is the utility of requirement, a, and W_a is the weight of requirement, a. The best combination chosen should be combination with the highest calculated utility, because the highest utility criterion was better at predicting the model-free combination then the equal utility criterion. There should be one combination of requirements for each category.

d The analyst should draft the final requirements set based on applying the Utility Curves Model. These requirements are from the winning combinations that were calculated in the previous step. This requirement set will be compared with the final requirements set of the Win-Win Model.

Steps 3 to 9 are part of the collaborative process:

Step 3 The first step of the Cost-Benefit Analysis Method (CBAM) is employed. In this step each stakeholder team determines the quality attributes or objectives of the system and quantify them as QAScores. Each stakeholder group will have a set of objectives from their own point of view. Each objective should have a QAScore based on importance such that the sum of all QAScores is 100. Section 2.3.2 has more information about QAScores., which is part of the Cost-Benefit Analysis Method (CBAM). The QAScores will be used in steps 7

and 8 when calculating the optimal solution.

Step 4 In this step, each stakeholder team individually identifies conflicts and lists these conflicts. These conflicts should be differences between the initial requirements of each stakeholder. Stakeholder groups should err on the side of listing too many conflicts rather than listing too few conflicts. That is to say if a stakeholder is unsure of whether or not to include a conflict in the list, it should be included. As mentioned in the Win-Win Model case study the stakeholder groups identified many conflicts, including definition clarification, misunderstood meaning, and conflicts meant for the design stage. This step is repeated if there are unresolved issues that need to be readdressed or if new conflicts arise after the first iteration. Steps 4 through 9 should repeat until all conflicts have been resolved.

Step 5 For every actor-oriented conflict, the analyst and the stakeholders should assess whether or not to use the ***i**** framework to visualize the conflict. It should be decided whether an ***i**** visualization will contribute to finding a solution to the conflict. The framework should be used when it is a question of defining the functions of the system users or components. Once it has been assessed that using the ***i**** framework is viable, the analyst should create a visualization of the situation. An actor-based conflicts involve the definition or functionality of

users of the software system being developed. If the conflict is not actor-oriented this step should be omitted.

Step 6 For every conflict the stakeholder groups involved should meet in a negotiation session and draft some possible resolution options to solve the conflict. There should be at least two resolutions to be evaluated. These resolution options will be evaluated using two different methods, CBAM and Utility Curves Method. It is possible that a solution to the conflict is found in negotiation session in this step; in that case the procedure can skip to step 9.

Step 7 In these next two substeps, CBAM is employed to choose the optimal solution from the resolution options found in the previous step.

a For each resolution option, each stakeholder group should assign a contribution value to the QAScores determined in Step 2. The contribution value should be a number between -1 and +1 for each quality attribute (objective). Where +1 is a positive contribution, while -1 is a negative contribution.

b Using the cost-benefit analysis method formulas, Equations 3.3 and 3.4, the best resolution should be chosen. The best resolution is the one with the highest overall desirability.

Step 8 The Utility Curves Model is used to find the best solution among the resolution

options drafted in Step 6. The model was adapted to be used to select the best resolution option and that was described in the second complementary process CP.2.

a For every resolution option item each stakeholder should assign a utility and a weight value. As described in the complementary process CP.2, the stakeholders do not need to assign new values, the utility is the contribution to each quality attribute plus 1 and the weight is the QAScore determined in step 7a.

b The best solution is calculated using the weighted average formula employed in the Utility Curves Model. The weighted average formula modified for use in the Win-Win Model is in Equation 4.1. It should be noted that one needs to add 1 to the contribution value to avoid dealing with negative utility values.

Step 9 The stakeholders should meet in another negotiation session and must adopt a winning solution based on the calculations made in Step 7 and 8. There will be at most two solutions presented by the two methods. With this information the stakeholders should be able to adopt a solution. If a solution is not adopted for this conflict, this conflict should be reidentified in the process moves back to Step 4.

Step 10 The analyst drafts the final requirements set based on applying the Win-Win Model. These requirements are a compilation of the conflict resolutions that were adopted in Step 9. This requirement set will be compared with the final requirements set of the Utility Curves Model.

Step 11 The final requirement set from the Utility Curves model should be compared to that list the requirements produced by the Win-Win Model. The stakeholders should meet in a negotiation session to discuss the issues. The final requirements set from both models would mostly be similar and would be adopted quickly. This session gives stakeholders an opportunity to discuss and reconcile differences in the two requirements set. Any other outstanding issues would also be discussed in this session.

Step 12 The final product of the hybrid model is a finalized requirement set that meets the approval of all stakeholders involved. This requirement set may be used in writing the specification of a software system being developed.

This hybrid model was based on leveraging the strengths of the individual models and addressing the limitations each model may have. The belief is that using the hybrid model is beneficial then using any one model individually. In future studies the claim that hybrid model works better needs to be validated in some sort of experimental or field study.

4.4.3 Limitations of the Hybrid Model

The hybrid model has many limitations; many of them have been inherited from the three individual models. We have also identified some limitations created by the hybrid model itself. The main limitation of the three individual models was that they were unable to automatically identify conflicts. This weakness was passed on to the hybrid. The issue of scalability limited the individual models. We did not know how the models behave when the context has requirements conflicts that number in the hundreds or higher. The models worked with the context had about 30 conflicts. We presume this scalability limitation extends to the hybrid model. We are also uncertain on how the individual models and therefore the hybrid model behaves when there are more than two stakeholders involved in the negotiations.

The main limitation inherent only to the hybrid model is that the model does not supply one single set of solutions but rather two solution sets. As the Utility Curves model and the Win-Win model each were able to supply a final solution, the hybrid does not. If the stakeholders are not able to negotiate a final solution based on the two solutions, the hybrid model is applied again, which does not guarantee success.

4.5 Summary

This chapter described all the findings of the research in this thesis. In the first part of this chapter the insights gained from each model is discussed. These insights are in the context of the three methodology questions. Also, the model-applied results of the case study is compared with the model-free results. The limitations and weaknesses of each model is and detailed.

In the second part of this chapter, the three models working in tandem is described. Since each model is limited in some areas, another model can complement it in the areas of weakness. The description of the hybrid model that takes into account the complementary processes is written. The three conflict resolution models able to work in tandem as a hybrid model is the main findings of this research.

Chapter 5

Conclusion

In this chapter the conclusions and contributions of the research are presented. In Section 5.1, the initial objectives of our research are reviewed. In Section 5.2, we will briefly overview the research in terms of the contribution. The impact our research will have in the Software Engineering academic community is discussed in Section 5.3. Section 5.4 contains the direction of future research. This is followed by some final remarks.

5.1 Review Of Objectives

A cursory survey of the software industry reveals that many software projects fall short of expectations or fail completely. Authors, including Faulk[5], Easterbrook[4] and Brooks[3], attribute these deficiencies the lack of effort exerted to understand

the software requirements in the early planning stages. According to Easterbrook[1] in the multiple stakeholder in development environment requirements need to be addressed as they often come into conflict, because different stakeholders have divergent goals and perspectives. It is important to address requirements conflicts in the early planning stages because Boehm[2] discovered that fixing requirements conflicts later is more expensive than when they are fixed early.

The first objective of the research herein is to search and analyze suitable models and tools that aid stakeholders collaboratively resolve requirements conflicts in the early planning stages. Essentially, we wanted to find models that can **identify, visualize, and solve** requirements conflicts.

The second objective was to apply the existing models in different contexts outside the initial context, for which the models were developed. The models were applied to a new context via a case study, where the models' ability to **identify, visualize, and solve** was examined.

5.2 Overview Of Research

This section will overview the main contributions of this work. The contributions were:

1. A critical performance evaluation of existing conflict resolution models.

2. A hybrid model that leverages the strengths and addresses the limitations of the models examined in 1.

5.2.1 Evaluation of Existing Models

The first step to achieve this contribution was to study existing conflict resolution models in literature. We reviewed five models that address requirements conflicts, namely Utility Curves Model[26], Win-Win Model[29], ***i**** Framework[34], NeMo-CoSe[38] and Viewpoints[36]. Each model was demonstrated to solve requirements conflicts in the context they were presented.

The second step was applying the models to a context different from which they were presented in order to gain a better understanding. Three of the models, namely Utility Curves Model, Win-Win Model and ***i**** Framework, were applied. We decided to apply the three models to a new context via a case study. The data was compiled as extensive video footage and some text.

The case study involved eliciting conflicts from the video footage and apply each model to the set of conflicts. Three basic methodology questions were asked for each model that was applied. The answers to the methodology questions are summarized in Table 5.1.

In the final step, we analyzed the results of the case study to elicit new insights of the models. We discovered strengths, weaknesses and other qualities of the models

Table 5.1: Case Study Question Answered

Model	Utility Curves	Win-Win	*i** Framework
Does the model characterize conflicts?	no	yes	yes
Does the model visualize conflicts?	no	no	yes
Does model generate solution?	yes	yes	no

that were not discussed in previous literature. Based on the analysis we decided that it is best that the models work together in tandem, which led to the second contribution of a hybrid model.

5.2.2 Hybrid Model

The first step in creating a hybrid model, in Section 4.4, was to understand the areas of strength and weakness for each individual model. In the second step we identified complementary processes where one model's limitation in one area can be covered by another model's strength in that same area. In the final step in the hybrid model was constructed based on the complementary processes identified in the previous step.

5.3 Impact of Contributions

In this section we will discuss how the main contributions of this research will be useful to the software engineering academic community. We beleive that the contributions will foster more interest in conflict resolution as it applies to requirements engineering. The following is the impact of each individual contribution:

5.3.1 Evaluation of Existing Models

The evaluation adds to the body of knowledge about the three models evaluated in this research. The new insights gained are from applying the models in a context different from the one in which they originally appeared. Therefore, the insights gained and the limitations discovered will provide more information for researchers interested in conflict resolution.

5.3.2 Hybrid Model

We hope that the hybrid model will be demonstrated and validated by other software engineering academics. It is expected that the hybrid model will evolve as it is tested in various different contexts. The hybrid model is not static, its processes will change as it is further validated. Some components may be removed and some components may be added from other existing conflict resolution models. Ultimately a validated hybrid model should be tested in industry until it becomes a standard

conflict resolution technique.

5.4 Future Research

The research presented in this thesis is a small piece of the broader conflict resolution puzzle. More research is required to complete this puzzle. We have identified some areas where further research would be beneficial. The research areas are:

1. The hybrid model described in Section 4.4 should be validated. The validation should ideally be done using different data sets and conducted as a case study.

2. Once the hybrid model is validated it should be demonstrated in a field study in industry. The field study will involve the actual stakeholders using the hybrid model in real-time to resolve requirements conflicts that arise in requirements negotiation. Ultimately the field study will demonstrate that the software project will be more successful when the hybrid model is applied to requirements engineering.

3. Requirements elicitation methods should be researched. The models used in this research use requirements that have already been elicited from the stakeholders. Elicitation techniques should be evaluated in order to see if conflicts can be reduced.

4. We were not able to study the full potential of the ***i**** framework because it operated under the more limited Strategic Dependency model. Only a limited number of conflicts could be visualized. One should apply ***i**** under the Strategic Rationale model and examine whether the scope can increase and more conflicts could be visualized.

5.5 Final Remarks

We came into this research wanting to address the problem of software project failure. Too many projects do not meet cost, time and quality expectations. Early literature review revealed that this problem is attributed to the fact that conflicts between requirements are not addressed in the planning stages. We sought models that solve requirements conflicts and applied the models to a context different from which they originally appeared. The application revealed strengths and limitations of each model. We created a hybrid model of all three models that leverages the strength and addresses the deficiencies of the individual models. We beleive that this hybrid model is a significant step towards reducing software project failure.

Bibliography

[1] S. M. Easterbrook, "Handling conflict between domain descriptions with computer-supported negotiation," *Knowledge Acquisition: An International Journal*, vol. 3, pp. 255–289, 1991.

[2] B. Boehm, *Software Engineering Economics*. Prentice-Hall, 1981.

[3] F. P. Brooks, "No silver bullet: Essence and accidents of software engineering," *IEEE Computer*, vol. 20, no. 4, pp. 10–19, 1987.

[4] B. Nuseibeh and S. M. Easterbrook, *Fundamentals of Requirements Engineering*. Addison-Wesley, 2007.

[5] S. R. Faulk, "Software requirements: A tutorial," in *Software Requirements Engineering 2nd Edition* (R. Thayer and M. Dorfman, eds.), pp. 1–22, IEEE Computer Society Press, 1994.

[6] H. Mifflin, ed., *The American Heritage Dictionary of the English Language, Fourth Edition*. Houghton Mifflin Company, 2004.

[7] D. of Defense, ed., *Dictionary of Military and Associated Words*. US Department of Defense, 2003.

[8] E. Britannica, ed., *Britannica Concise Encyclopedia*. Encyclopdia Britannica, Inc., 2003.

[9] J. W. Burton, "Conflict resolution as a political system," *The Institute for Conflict Analysis and Resolution Working Papers*, 1993.

[10] W. N. Robinson and S. Fickas, "Supporting multi-perspective requirements engineering," in *Proceedings of the First International Conference on Requirements Engineering*, pp. 206–215, 1994.

[11] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Software*, vol. 12, no. 5, pp. 52–62, 1995.

[12] M. Bekker, J. Beusmansb, D. Keysonb, and P. Lloyd, "Kidreporter: a user requirements gathering technique for designing with children," *Interacting with Computers*, vol. 15, pp. 187–202, 2003.

[13] B. Boehm, H. In, T. Rodgers, and M. Deutsch, "Applying win-win to quality requirements: A case study," in *Proceedings of the 23rd International Conference on Software Engineering*, pp. 555–564, 2001.

[14] M. Jackson, "The meaning of requirements," *Annals of Software Engineering Special Issue on Software Requirements Engineering*, pp. 5–22, 1997.

[15] F. P. Brooks, *The Mythical Man-Month.* Addison-Wesley, 1975.

[16] "Requirements engineering specialist group." World Wide Web site. http://www.resg.org.uk/.

[17] B. Nuseibeh and S. M. Easterbrook, "Requirements engineering: A roadmap," in *Future of Software Engineering* (A. C. W. Finkelstein, ed.), IEEE Computer Society Press, 2000.

[18] B. Boehm, P. Bose, E. Horowitz, and M. J. Lee, "Software requirements as negotiated win conditions," in *Proceedings of the First International Conference on Requirements Engineering*, pp. 74–83, 1994.

[19] B. Boehm and R. Ross, "Theory w software project management: Principles and examples," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 902–916, 1989.

[20] G. Kotonya and I. Sommerville, "Requirements engineering with viewpoints," *Software Engineering Journal*, vol. 11, no. 1, pp. 5–18, 1996.

[21] O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the First International Conference on Requirements Engineering*, pp. 94–101, 1994.

[22] H. In, T. Rodgers, and D. Olson, "Multi-criteria preference analysis for systematic requirements negotiation," in *Proceedings of the 26th Annual International Computer Software and Applications Conference*, 2002.

[23] M. Deutsch, *The Resolution of Conflict*. Yale University Press, 1973.

[24] S. P. Robbins, *Managing Organizational Conflict: A Non-traditional Approach*. Prentice Hall, 1974.

[25] A. Strauss, *Negotiations: Varieties, Contexts, Processes and Social Order*. Jossey-Bass Publishers, 1978.

[26] W. N. Robinson, "Negotiation behavior during requirement specification," in *Proceedings of the 12th International Conference on Software Engineering*, pp. 268–276, 1990.

[27] E. Kettunen, "Interactive methods for group decision and negotiation support." This is a Licentiate's Thesis, January 1999.

[28] B. Boehm, "A spiral model of software development and enhancement," *IEEE Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[29] B. Boehm, P. Bose, E. Horowitz, and M. J. Lee, "Software requirements negotiation and renegotiation aids: A theory-w based spiral approach," in *Proceedings of the 17th International Conference on Software Engineering*, pp. 243–253, 1995.

[30] H. In, R. Kazman, and D. Olson, "From requirements negotiation to software architectural decisions," in *Proceedings of the First International Workshop From Software Requirements to Architectures*, 2001.

[31] B. Boehm and H. In, "Identifiing quality-requirement conflicts," *IEEE Software*, vol. 13, no. 2, pp. 25–35, 1996.

[32] B. Boehm and H. In, "Software cost option strategytool (s-cost)," in *Proceedings of the 20th Annual International Computer Software and Applications Conference*, pp. 15–20, 1996.

[33] R. Kazman, J. Asundi, and M. Klein, "Quantifying the costs and benefits of architectural decisions," in *Proceedings of the 23rd International Conference on Software Engineering*, pp. 297–306, 2001.

[34] E. S. K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pp. 226–235, 1997.

[35] S. Easterbrook, E. Yu, J. Aranda, Y. Fan, J. Horkoff, M. Leica, and R. A. Qadir, "Do viewpoints lead to better conceptual models? an exploratory case study," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pp. 199–208, 2005.

[36] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, "Viewpoints: A framework for integrating multiple perspectives in system development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, pp. 31–58, 1992.

[37] S. Easterbrook and B. Nuseibeh, "Managing inconsistencies in an evolving specification," in *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, pp. 48–55, 1995.

[38] T. Wanyama and B. H. Far, "A multi-agent framework for conflict analysis and negotiation: Case of cots selection," *IEICE Transactions on Information and Systems*, vol. 88, no. 9, pp. 2047–2058, 2005.

[39] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Software*, vol. 14, no. 5, pp. 67–74, 1997.

[40] D. Cubranic, "Csc 576b - global software development - course website." World Wide Web site. http://segal.cs.uvic.ca/csc576b/.

[41] D. Damian, A. Scott, L. Izquierdo, S. Gupta, and R. Elves, "Leveraging design patterns in global software development," in *Proceedings of the International Workshop on Distributed Software Engineering*, 2005.

Appendix A

Requirements Specification Document

This chapter contains excerpts from the Requirement Specification document that describes the Media Distribution Platform designed in the case study.

Software Requirements Specification

1. Introduction

1.1 Purpose

This document describes the system requirements for a media distribution platform system, including content management, content delivery, consumer data management, and security systems. It is intended for the Vegemite Kids group, the clients for this project, as well as selected members of Media Magic Inc, including the de-

signers, developers, and testers for the system.

1.2 Scope

The software system to be produced is the Media Distribution Platform System. Media Distribution Platform is intended to provide an efficient process to distribute film trails via bulk emails in order to generate public interest targeted to specific consumer groups.

....

2. Users

There are five primary types of end users.

2.1 Marketing employees

Marketing Employees will be able to use the Media Distribution Platform to perform many of their marketing/advertising activities from their interface.

....

2.3 Marketing Managers

Marketing Managers are responsible for administrative roles.

....

2.5 Developers/Testers

Developers/Testers will be provided with limited access to enable them to evaluate and develop the system. Their main function will be to ensure that all the requirements are met according the specifications provided in this Requirements Specification

document.

....

3.1 Functionality

3.1.1 Consumer Data Management System

3.1.1.1 Bulk Import of consumer email lists Authorized employees and clients shall be able to import lists of consumers (existing and new), in order to eliminate the need to transfer data into the local database manually. In addition, authorized employees must be able to import lists of no-spam addresses that are not to be targeted by any email distribution activity. This functionality is required in order to maintain compliance with US anti-spam law.

....

3.1.2 Content Management System

3.1.2.1 Add/Edit/Remove content

Marketing Employees shall be able to add, edit, and remove content sets using the user interface. This is in order to facilitate new content development.

....

3.1.5 Security

3.1.5.1 Virus Scanning

The system shall check all content and attachments for viruses before delivery to the consumer. In the event that a virus infects the internal network, it is imperative

that the virus does not propagate through the bulk content distribution channels.

....

3.2 User Management

3.2.1 Graphical User Interface

The Media Distribution Platform system requires an interface for managing user accounts and respective access levels as outlined above (3.1.5). The interface must allow the Marketing Manager to create user accounts and easily assign access levels that the user has been authorized for. Additional functionality must also exist for listing, deleting or disabling user accounts.