The WinWin Requirements Negotiation System: a Model-Driven Approach

Mingjune Lee and Barry Boehm
Center for Software Engineering and Computer Science Department
University of Southern California
Los Angeles, CA 90089
Telephone: 213-740-9731, Fax: 213-740-4927
E-mail: <milee,boehm>@cs.usc.edu

Abstract

Requirements Engineering constitutes an important part of Software Engineering. The USC WinWin requirements negotiation system addresses critical issues in requirements engineering including (1) multi-stakeholder consideration, (2) change management, and (3) groupware support. This paper presents our current research efforts on constructing and reconciling several formal and semi-formal models of the system and its operations, including *inter-artifact relationship, artifact life cycles, and equilibrium model*. It concentrates on determining the relationships among the various models or views of the WinWin requirements engineering process.

Keywords: concurrent engineering, groupware, model-driven processes, negotiation, requirements engineering, spiral model, Theory W, win condition.

1 Introduction

Requirements Engineering (RE), which provides a systematic framework for representing and acquiring software requirements, constitutes an important part of Software Engineering. Research [NKF94, BR89, Dav90, T+96, EGR91] indicates that the following issues are critical to requirements engineering.

Multi-stakeholder considerations: In a collaborative software development environment, perspectives of all stakeholders should be integrated and reconciled. Past requirements engineering approaches focusing on user

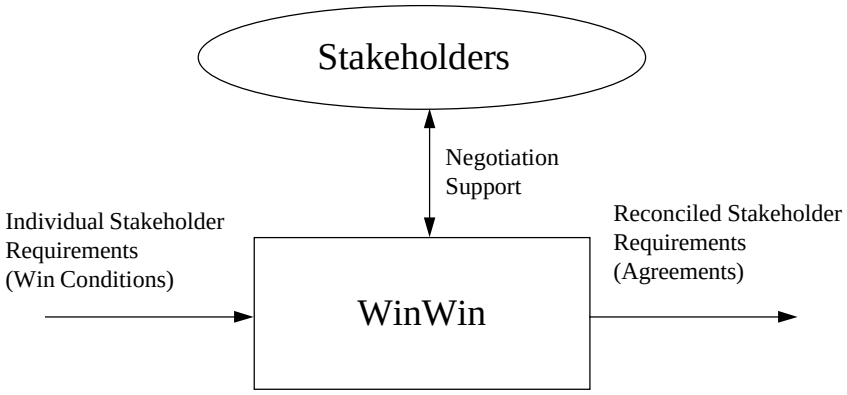

Figure 1: The WinWin requirements negotiation system overview

requirements failed in that users' perspectives may not match other stakeholders' perspectives and thus result in a win-lose situation. In addition, user requirements are often misinterpreted because other key stakeholders are not involved in the requirements specification phase.

Change management: It is very often that some criteria are changed in the midst of requirements formulation. For example, a budget cut would invalidate some previous agreements. Therefore, change management is necessary to accommodate changes in objectives, constraints, or alternatives. In addition, the rationale for previous requirements needs to be incorporated to help determine how to change requirements.

Groupware support: To facilitate requirements negotiation across organizations, groupware that institutionalizes *computer supported cooperative work (CSCW)* is needed to bridge the geographical and time gap as well as to achieve group decisions.

The WinWin requirements negotiation system[BBHL94b, BBHL95] takes in individual stakeholder requirements (win conditions) and help produce reconciled stakeholder requirements (agreements), as illustrated in Figure 1, to address *multi-stakeholder consideration*. In order to achieve this goal, "issues" are provided to capture sets of conflicting requirements and "options" are possible resolutions to "issues". The system, on the top level, is driven by the WinWin Spiral Process[BBHL95] to accommodate *change* of requirements. In addition, the WinWin artifacts described in Section 2 keep track of the rationale needed and the WinWin equilibrium model facilitates the process. As *groupware*, the system provides negotiation support, message passing and multi-media document navigation.

The development approach for the USC WinWin System to date has primarily involved exploratory prototyping. Several experiments[BBHL94a, Buc94] demonstrated that the WinWin System facilitates the requirements negotiation. The system is now converging on a relatively stable set of artifacts and relationships. This makes it feasible and important to elaborate the underlying process model and to formalize these artifacts and relationships to provide a solid scientific framework for the WinWin system. This paper presents our current research efforts on constructing several formal and semi-formal models of the system and its operations[Lee95]. The big challenge here is to determine the relationships among the various views and reconcile them. We will also exhibit our approach of reconciling the many views of the WinWin system. Section 2 describes major WinWin artifacts and their intra/inter-relationships for supporting the requirements negotiation infrastructure. Section 3 shows how WinWin models the requirements negotiation dynamics by proposing the WinWin equilibrium model, WinWin artifact life cycles, and their formal definitions. Section 4 provides a framework for determining the relationships among the various models or views to reconcile them. Section 5 compares WinWin to related requirements engineering approaches. Section 6 summarizes the contributions made with this research and discusses future works.

2 Supporting the Requirements Negotiation Infrastructure

2.1 Problem Space View: the Inter-Win-Condition Relationships

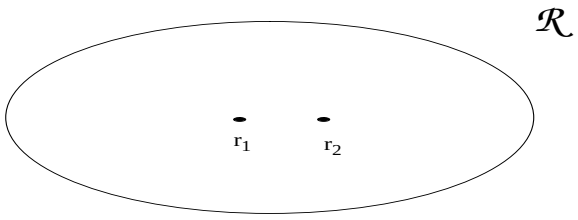

Figure 2: Requirements Space

Consider the space R of all requirements specifications r (see Figure 2). Each point r $\in$ R consists of a set of functional, performance, interface, and attribute specifications. For example, the only difference between r_1 and r_2 in Figure 2 may be that r_1's response time is 1 second and r_2's is 2 seconds.

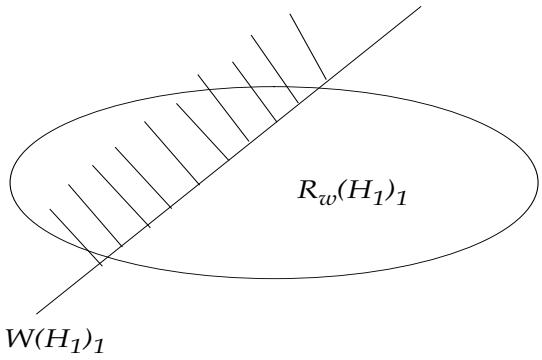

Figure 3: Requirements Space

We can then define a Win Condition as a constraint on R, dividing R into mutually exclusive subsets of requirements specifications which do or do not satisfy the Win Condition. Specifically, for the i^{th} stakeholder H_i, his/her j^{th} win condition $W(H_i)_j$ defines the following subset of R (see Figure 3):

$$R_w(H_i)_j = \{r : r\ satisfies\ W(H_i)_j\}.$$

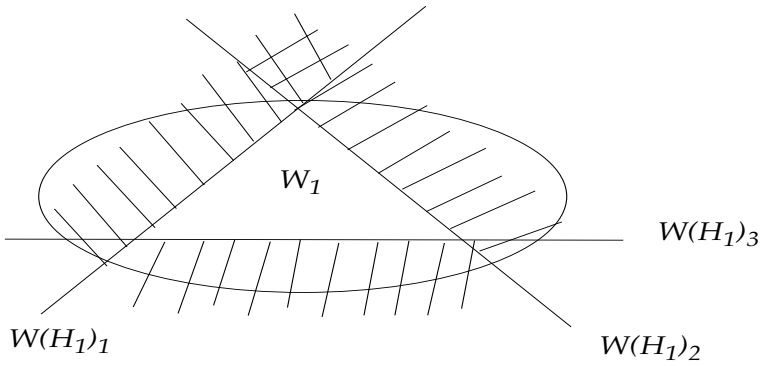

Figure 4: Requirements Space

For example, if $W(H_1)_1$ expressed stakeholder H_1's desire that the software cost be less than \$7 million, $R_w(H_1)_1$ would be the set of all requirements specifications which could be implemented for less than \$7M.

If stakeholder H_1 has n_1 win conditions, his/her win region W_1 can be expressed as the intersection of all of the individual win condition regions (see Figure 4):

$$W_1 = \bigcap_{j=1}^{n_1} R_w(H_1)_j.$$

By definition, any win conditions of stakeholder H_1 belongs to his/her win region W_1.

We can also use this framework to define the WinWin Issue, Option, and Agreement artifacts.

An agreement A_k covers a set of non-controversial win conditions whose win regions have a non-empty set.

Definition 2.1

$$A_k = \{W(H_i)_j | \bigcap_{i,j} R_w(H_i)_j \neq \phi\}.$$

An Issue I_k defines a minimal set of win conditions whose win regions have an empty intersection. To define I_k , we first define E to be the set of win conditions whose regions have an empty intersection.

$$E = \{W(H_i)_j | \bigcap_{i,j} R_w(H_i)_j = \phi\}.$$

The definition of E actually invites other irrelevant win conditions to be part of the issue. Once some win conditions constitute an issue (when the intersection of their win regions is empty), we can include other win conditions infinitely and the intersection of their win regions will remain empty. To avoid including irrelevant win conditions, we further define the minimal set Min_Set of E to include only win conditions which are the real sources of that issue. The minimal set means that if a win condition is taken from it, the intersection of the win regions of the remaining elements will no longer be empty.

$$Min\_Set(E) = \{w| \bigcap_{W(H_i)_j \in (E-\{w\})} R_w(H_i)_j \neq \phi\}$$

As sets that satisfy the Min_Set(E) definition are multiple, I_k is in fact one set that is chosen by the stakeholder.

Definition 2.2

$$I_k \in Min\_Set(E)$$

It can be shown that the win conditions in I_k must fall in the area outside of some other stakeholder's win region W_i:

$$\forall w \in I_k, \exists W_i \; s.t. \; w \notin W_i.$$

An example would be that a customer H_1 has a win condition w_{11} that the software cost be less than \$7M. And the user H_2 has win conditions $\{w_{21}, w_{22}, w_{23}\}$ specifying functions {Simulation,Reporting,Testing} whose costs are {\$4M,\$3M,\$5M}, respectively. Sets that satisfy E would be any supersets of $\{w_{11}, w_{21}, w_{22}, w_{23}$ $\}$. Whereas, the minimal sets are $\{w_{11}, w_{21}, w_{23}\}$ and $\{w_{11}, w_{22}, w_{23}$ $\}$. I_k can be either one according to the stakeholders' decision.

For an Issue I_k , an Option O_{kl} is a proposed relaxation $\{\triangle W(H_i)_j\}_l$ of the set of win conditions $\{$ $W(H_i)_j$ $\}$ constituting the Issue. The requirements space expanded for stakeholder H_1 by his/her options is called the satisfactory region S_1:

$$S_1 = \bigcap_{j=1}^{n_1} [R_{\triangle w}(H_1)_j]$$

A feasible Option is one which makes a non-empty intersection of the relaxed win conditions:

Definition 2.3

$$O_{kl} = \{\triangle w(H_i)_j | \bigcap_{i,j} [R_{w+\triangle w}(H_i)_j] \neq \phi.$$

The lose region L_1 of stakeholder H_1 is thus

$$\mathcal{R} - (W_1 \cup S_1)$$

or

$$\mathcal{R} - \bigcap_{i} [R_{w+\triangle w}(H_i)].$$

And it can be proved that an unresolvable Issue I_k must fall in at least one stakeholder's lose region:

$$\forall issue I_k, I_k \; is \; unresolvable \; iff \exists L_m \; s.t. \; I_k \subset L_m.$$

With respect to the example, some candidate options would be to increase the budget limit by $\triangle w(H_1)_1$ = \$1M; to defer some of the user functions $\{$ $\triangle w(H_2)_j$ $\}$; or various combinations of these.

An agreement A_k is then a resolution to resolve Issue I_k by adopting a feasible option O_{kl} , which is passed by all the stakeholders.

Definition 2.4

$$A_k = \{(w + \triangle w)(H_i)_j | \bigcap_{i,j} [R_{w+\triangle w}(H_i)_j] \neq \phi\}.$$

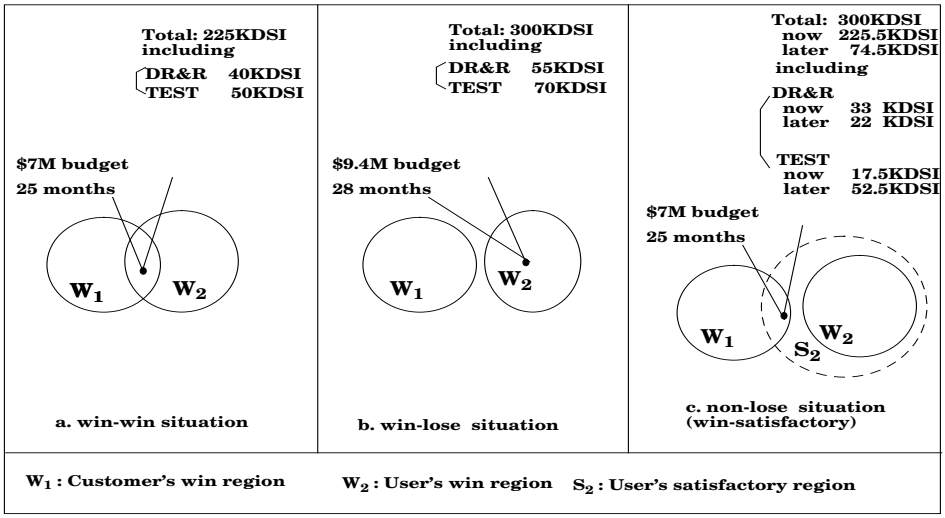

Figure 5: The inter-win-condition relationships (example)

Definition 2.1 is a special case of Definition 2.4, where $\triangle w$ is null. It can be shown that an agreement is contained by the acceptable regions, namely, {SS, WS, WW, SW} in the 2-stakeholder case.

The set definitions can be illustrated using set diagrams exemplified by Figure 5. W_1 and W_2 are Customer's and User's win regions, respectively. User at first presents win conditions for functionalities that require 225KDSI (thousand delivered source instructions), including 40KDSI for Data Reduction and Reporting (DR&R) and 50KDSI for TEST. This can be done within $7M (million) budget and 25 months. As it is in Customer's win region, it is a win-win situation. Due to change of requirements, User is asking for more functionalities that increase the code size to 300KDSI. The budget and the cost are thus raised beyond what Customer can supply. And now, it results in a win-lose situation.

One way to resolve this win-lose situation is to explore both stakeholders' satisfactory regions S_i's as options. In this example, User agrees to defer part of the DR&R and the TEST modules until more money becomes available. This implies that 225.5KDSI be implemented at this moment which again can be done within 25 months and $7M budget. It now settles in a non-lose (or win-satisfactory) situation.

A complete enumeration of relationships between {Win,Lose,Satisfactory} regions of two stakeholders is illustrated in Figure 6. Figure 7 is a projection of Figure 6, generalizing the 2-stakeholder case into the n-stakeholder one. The total number of regions is 3^n as it is equivalent to the number of a 3-letter

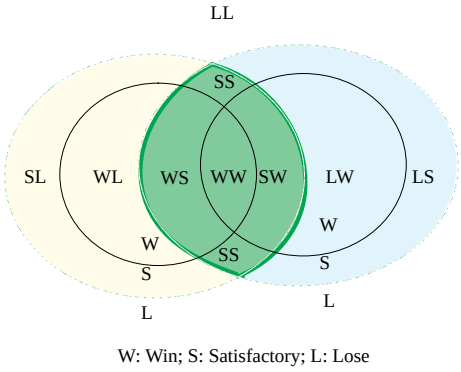

Figure 6: The inter-win-condition relationships

{W,S,L} permutation in n-digit where repeated letter is allowed. These regions are dichotomized into the following types:

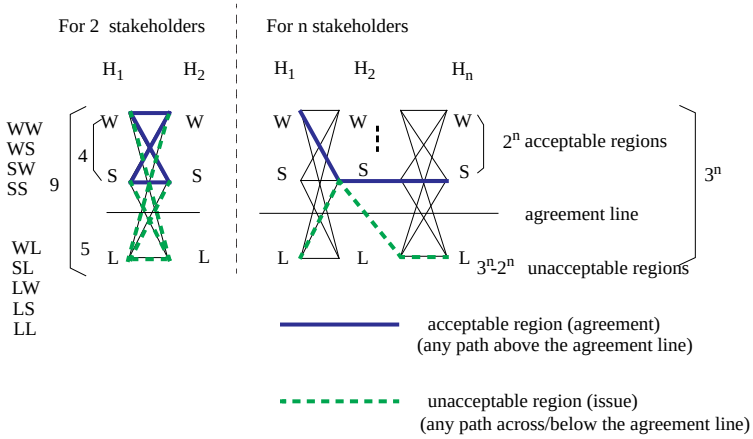

Figure 7: The generalized inter-win-condition relationships

- Acceptable region:
  This type of regions contain requirements that do not fall in any stakeholder's lose region. By definition, it proposes non-empty intersection between requirements and contends passed agreements. Any path above the agreement line satisfies this criteria as it contains only stakeholders'

win or satisfactory regions. The number of qualified regions is 2^n as it is equivalent to the number of a 2-letter {S,W} permutation in n-digit where repeated letter is allowed.

- Unacceptable region:
  This type of regions contain requirements that does fall in some stakeholder's lose region and results in unresolvable issues. Its number is $3^n(total) - 2^n(acceptable)$.

2.2 Major WinWin Artifact Types and Their Relationships

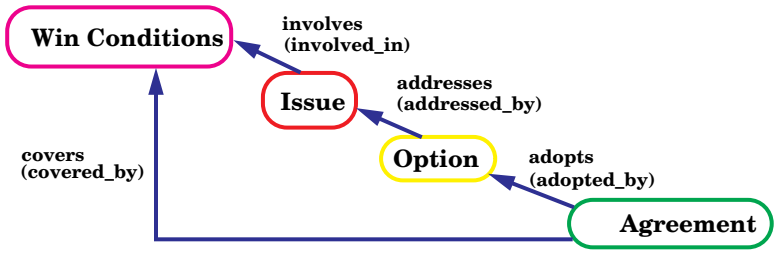

Figure 8: WinWin artifact relationships

As illustrated in Figure 1, the WinWin system takes *Win Conditions* as individual requirements and helps the stakeholders to obtain agreements as reconciled requirements. In order to facilitate this, Section sec-inter-win models the inter-win-condition relationship and defines 4 types of artifacts to support the negotiation infrastructure — *Win Condition*, *Issue*, *Option*, and *Agreement*. The relationships between different WinWin artifact types are portrayed in Figure 8. An issue *involves* many controversial win conditions. An option *addresses* an identified issue. An agreement either 1). directly *covers* a set of non-controversial win conditions or 2). *adopts* a feasible option to resolve the corresponding issue and covers all the win conditions involved in that issue.

The definition of the four types of artifacts with respect to the problem space view is summarized as follows:

Win Condition: a stakeholder's requirement that (s)he considers important and beneficial; as mentioned in the previous section. A stakeholders' win condition set defines his/her win region W_i.

Issue: an aggregate I_k that addresses conflicting win conditions; that is, a minimal set of win conditions whose win regions have an empty intersection. An issue I_k falls outside of some stakeholder's win region W_i. That is I_k must fall in some stakeholder's satisfactory or lose region(=$(S_i \cup L_i)$). For the 2-stakeholder case, it includes {WS,SW,SS,WL,LW,SL,LS,LL}. An unresolvable I_k falls in at least one stakeholder's lose region L_i; that is, the unacceptable regions in Figure 7. For the 2-stakeholder case, it includes {WL,LW,SL,LS,LL}. It is noted that the number of unresolvable issues are decreased by proposing options to explore the satisfactory region (S).

Option: an alternative that is proposed to resolve an issue. For an Issue I_k , an Option O_k is a proposed relaxation { $\triangle W(H_i)_j$ } of the set of win conditions { $W(H_i)_j$ } constituting the Issue. It extends the win region (W) to the satisfactory (S) region in Figure 6. A feasible Option is one which has a non-empty intersection of the relaxed win conditions.

Agreement: an aggregate that captures reconciled win conditions either by directly covering non-controversial win conditions or adopting an feasible option to an issue that involve multiple controversial win conditions. By definition, it is only contained by the acceptable regions as shown in Figure 7.

Given the definitions of artifacts and their valid relationships, an artifact chain is a graph of artifact nodes connected by valid relationship links. Figure 9 shows several examples of artifact chains.

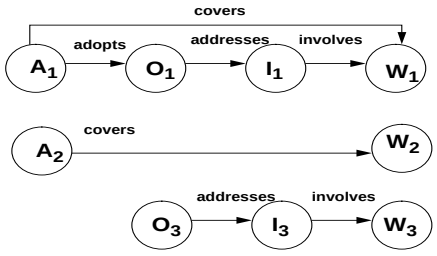

Figure 9: Artifact chain examples

An artifact chain is defined as:

Definition 2.5

$$
\begin{aligned}
chain &::= [I-chain | A_c - chain] < Win\ Condition > \\
I-chain &::= [O-chain] < Issue > \stackrel{involves}{\longrightarrow} \\
O-chain &::= [A-chain] < Option > \stackrel{addresses}{\longrightarrow} \\
A_a - chain &::= < Agreement > \stackrel{adopts}{\longrightarrow} \\
A_c - chain &::= < Agreement > \stackrel{covers}{\longrightarrow}
\end{aligned}
$$

3 Modeling the Requirements Negotiation Dynamics

In this section, the WinWin Equilibrium for modeling the requirements negotiation dynamics is elaborated in hierarchical state charts and in predicate calculus. The complementary life cycle model of WinWin artifacts demonstrate how artifacts evolve in the WinWin Equilibrium.

3.1 The WinWin Equilibrium Model

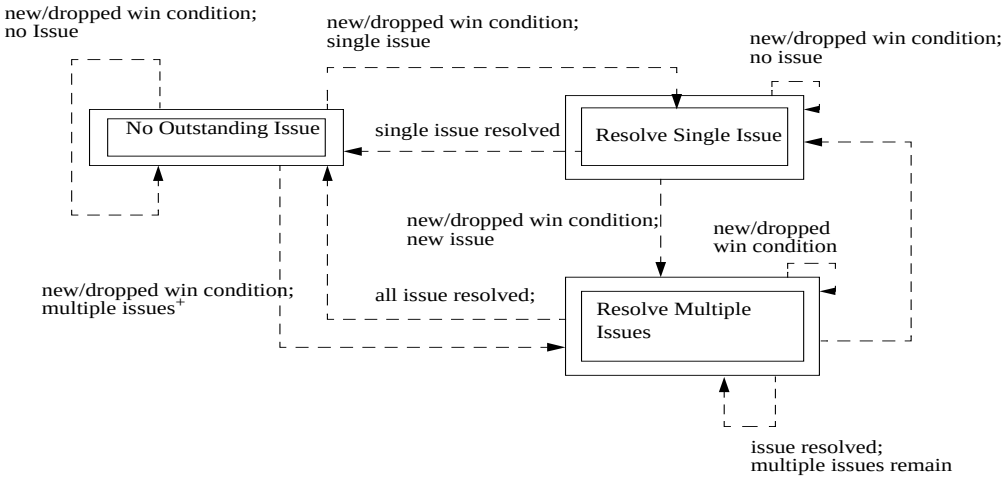

Figure 10: Top Level of the WinWin Equilibrium Model

The WinWin Equilibrium Model is an elaboration of the WinWin Spiral Process Model to guide the stakeholders through the process. In the negotiation and renegotiation process, all stakeholders work toward achieving the WinWin

Equilibrium state, in which all issues are resolved. and all active win conditions are covered by some passed agreements.

Figure 10 shows the top level of the WinWin Equilibrium Model. It is characterized by:

1. no outstanding issue:

$$\nexists I_k$$

2. Resolve single issue:

$$\exists!(there\ exists\ only\ one) I_k.$$

3. Resolve multiple issues:

$$\exists I_1, I_2\ s.t.\ I_1 \neq I_2.$$

When a new win condition comes in the "no outstanding issue" state and does not generate any new issue, the system will stay in "No outstanding Issue". It can transit from "No outstanding Issue" to "Resolve Single Issue" when it conflicts with some existing win conditions and generates one issue. In like manner, it can transit to "Resolve Multiple Issues" when this new condition generates more than one issues. "Resolve Single Issue" and "Resolve Multiple Issues" need to be discussed separately in that resolving multiple issues requires considering inter-locking issues. In [BBHL95], a model for recovering the WinWin equilibrium from a single issue was presented. As in the real world, stakeholders are facing multiple and inter-locking issues. The hierarchical equilibrium model proposed here is our subsequent work to cover the more complex multi-issue case.

Figure 11 shows the sub-states inside state "No outstanding Issue", where the WinWin Equilibrium state is part of. The definition of a WinWin Equilibrium state is an example of how each state is formalized in the WinWin Equilibrium Model:

$$\bigcap_{\forall i, \forall j} [R_w(H_i)_j \neq \phi].$$

The WinWin Equilibrium state can be unstabilized when a new win condition is entered. All stakeholders assess the new win condition and decide whether it raises any conflict. If no apparent conflicts are found, stakeholders can propose this new win condition to be an open agreement that waits for each stakeholders to commit to this agreement by voting (and this commitment will turn the open agreement into a passed one). Otherwise, an issue will be posted to address a conflict and options will be proposed to resolve this issue. Stakeholders now compare the many options to their win conditions of high priority as well as existing agreements to filter out options that are not feasible. The surviving options then will be compared to each other with the aide of trade-off

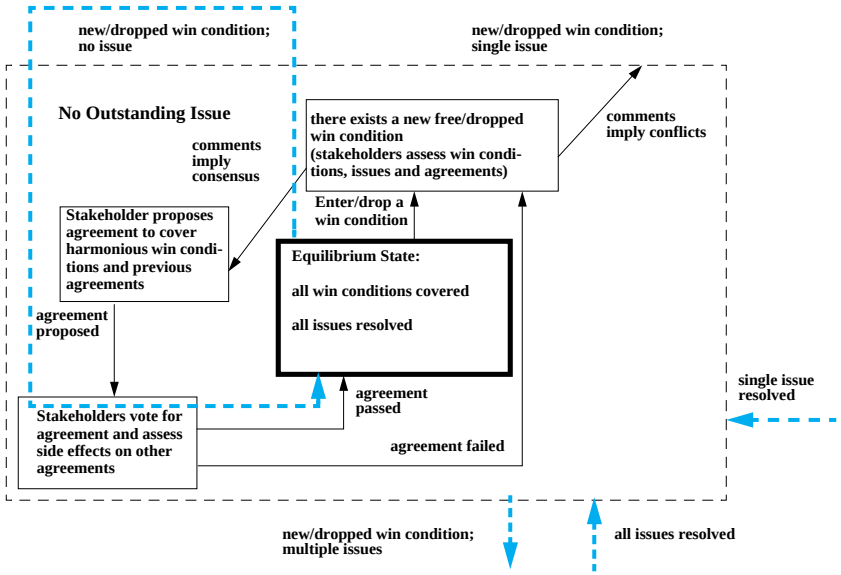

Figure 11: Sub-states in "No outstanding Issue"

analysis tools to negotiate an optimal option. Upon achieving a consensus, an open agreement will be composed based on the adopted option. Stakeholders now examine wordings of the new open agreement and decide whether it can be passed or it conflicts with other passed agreements and requires another run of conflict resolution.

3.2 The WinWin Artifact Life Cycles

The WinWin artifact life cycles shows how the artifacts evolve in the negotiation process. Before the formal analysis, Figure 12 presents life cycle of a agreement. By the formal analysis, we realize that this over-simplified life cycle provides very little information to the WinWin users. They really need a life cycle on the artifact that indicates

- how other artifacts are related to this particular artifact and (inter-artifact relationships)
- how far the artifact is from contributing to reaching consensus among stakeholders (equilibrium model and inter-artifact relationship)

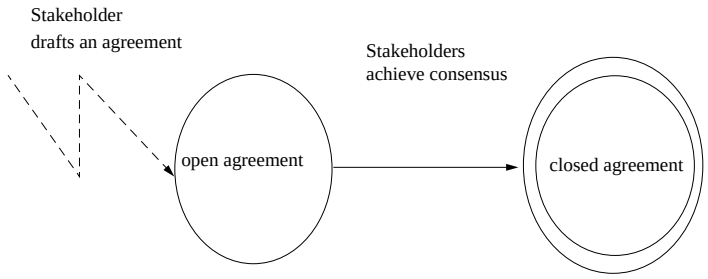

Figure 12: Life cycle of agreement(before analysis)

- whether any action needs to be taken to facilitate negotiation (equilibrium model)

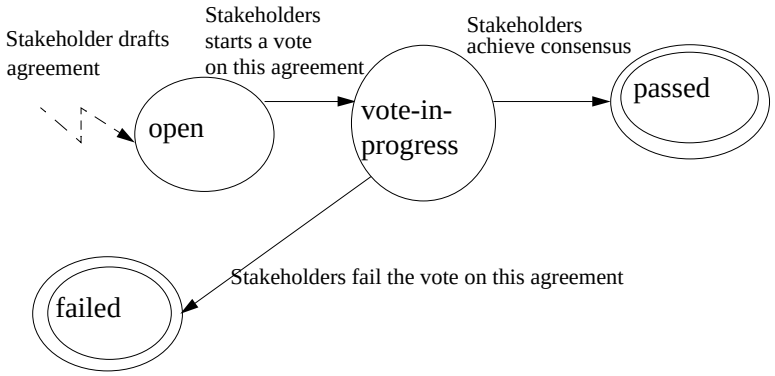

Figure 13: Life cycle of agreement(after analysis)

Consider these criteria, we determine to implement the agreement artifact life cycle as a function reflecting states in the Equilibrium model and the Inter-artifact relationships. That is, for every state in the Equilibrium model, the agreement has a defined state according to their inter-artifact relationship. The other artifact life cycles are functions determined by agreement. With this definition, we also find that the Artifact Life Cycles connect the Equilibrium Model and the Artifact Relationships. And since artifacts and their relationships

are actually functions defined by the inter-win-condition relationships, the many views of WinWin can be consolidated.

The model as shown in Figure 13 is a revised version after our analysis. Each state here is noted by the "state" field in the agreement schema. An agreement comes into being either by covering a set of non-controversial win conditions or adopting a selected optimal option for resolving conflicts. Its initial status is always *open*. In the equilibrium model, it instructs stakeholders to use voting mechanism to express their option (concur, don't concur, abstain) on this agreement. Once a vote is conducted on an agreement, the state is then turned into *vote-in-progress*. The text of all artifacts referenced by it (directly or indirectly) will be locked during this voting process. When stakeholders pass the vote on this agreement, its state is changed into "passed" and the text of all its related artifacts will remain locked. If stakeholders fail the vote on this agreement, the agreement will be inactive and all its referenced artifacts will be unlocked for further negotiation.

The states for an option life cycle is {unused, pre-used, vote-in-progress, used}. It is defined according to the state of the agreement that adopts it. *Unused* means that it is not adopted by any agreement. *Pre-used* shows it is adopted by an open agreement. *vote-in-progress* indicates its adopting agreement is now being conducted a vote. *Used* implies that the adopting agreement is passed. This life cycle points out whether it has an established relationship with an agreement, how far it is from reaching consensus and whether it is locked. Formal definition examples will be demonstrated in the following section. Details of other WinWin artifact life cycle models are studied and presented in [Lee96].

3.3 The WinWin State Definition Example

After life cycles for each artifact in the WinWin system have been studied and completed, the following set relations and predicate calculus furnish formal definitions for states in Figure 10.

3.3.1 WinWin Artifact Set Definitions

W: set of win conditions

I: set of issues

O: set of options

A: set of agreements

Define unary function state(x) for the artifact life cycle model:

$$
\begin{array}{lcl}
W & \rightarrow & \{free, uncovered, pre-covered, vote-in-progress, covered\} \\
I & \rightarrow & \{unresolved, addressed, pre-resolved, vote-in-progress, resolved\} \\
O & \rightarrow & \{unused, pre-used, vote-in-progress, used\} \\
A & \rightarrow & \{open, vote-in-progress, passed, failed\}
\end{array}
$$

Define unary function status(x):

$$(W \cup I \cup O \cup A) \rightarrow \{active, inactive\}$$

Relationships between different artifact types discussed in Section 2.2 are denoted by the following *multiple-valued* functions:

$$\begin{aligned} addresses(x): & \quad O \rightarrow I \\ adopts(x): & \quad A \rightarrow O \\ involves(x): & \quad I \rightarrow W \\ covers(x): & \quad A \rightarrow W \end{aligned}$$

3.3.2 WinWin Equilibrium State

The negotiation process is in the WinWin equilibrium state if the following are all satisfied:

$$(\forall w \in W | (state(w) = covered)) \wedge$$
$$(\forall i \in I | (state(i) = resolved))$$

4 An Integrated Model

Sections 2 and 3 present the many views of the WinWin requirements engineering process. The goal of this section is to describe the relationships between many views to establish an integrated model that is consistent and expressive.

As described in Section 2, the problem space view models the inter-win-condition relationship and suggests the 4 types of WinWin artifacts with their relationships to cover non-controversial and reconcile controversial win conditions into passed agreements.

As noted before, the states in the artifact life cycles is a function of the current negotiation state indicated by the Equilibrium model as well as the inter-artifact relationships. Section 2.2 literally described how the function is defined. The following is an instance how the states can be formalized.

For an issue i, state(i) is

- **unresolved:**

$$iff \quad \nexists o \in O \; s.t. \; (i \in addresses(o))$$

- **addressed:**

$$iff \; \exists o \in O \; s.t. \; (i \in addresses(o)) \wedge (state(o) = unused)$$

- **pre-resolved:**

$$iff \; \exists o \in O \; s.t. \; (i \in addresses(o)) \wedge (state(o) = pre - used)$$

- **vote-in-progress:**

$$iff\ \exists o \in O\ s.t.\ (i \in addresses(o)) \wedge (state(o) = vote - in - progress)$$

- **resolved:**

$$iff\ \exists o \in O\ s.t.\ (i \in addresses(o)) \wedge (state(o) = used)$$

Artifact Type	State						
Win Condition	free	uncovered	uncovered	pre-covered	vote-in-progress	covered	uncovered
Issue		unresolved	addressed	pre-resolved	vote-in-progress	resolved	addressed
Option			unused	pre-used	vote-in-progress	used	unused
Agreement				open	vote-in-progress	passed	failed

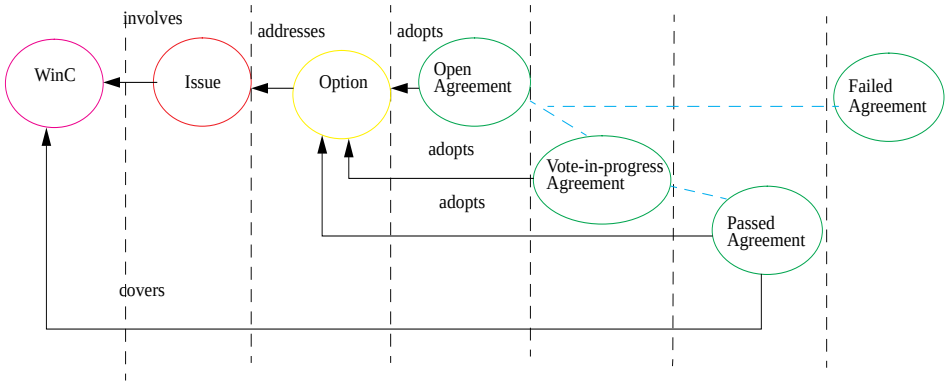

Figure 14: Artifact Life Cycle v.s. Artifact Relationships

Figure 14 demonstrates an example of how the state of a win condition evolves with respect to other artifacts that it relates to:

1. when it is just created,
2. when it is involved in an unresolved issue,
3. when the involving issues was addressed by options,
4. when some addressing option is adopted by an open agreement,

5. when that adopting agreement is in-progress (while a vote is being conducted by the stakeholders

6. when the adopting agreement is voted as passed

7. when that adopting agreement is voted as failed.

For clear illustration, this example intentionally assumes that the win condition is involved in only one issue. A complete enumeration that covers cases when a win condition is involved in multiple issues and/or covered by multiple agreements can be found in [Lee96].

The state of an artifact shows for the user how far it is from contributing to the equilibrium state. For example, if an agreement is in the state of open, it still needs to be assessed and discussed in order to start a vote. Once it is passed, it indicates that stakeholders have consensus on this agreement and all its covering win conditions are closer to be reconciled.

While Problem Space are identified and presented by the WinWin Artifact and their relationships. These Artifacts Life cycles, defined as a function reflecting both the inter-artifact relationship and the negotiation state, connect these relationships with the equilibrium model. Every state in the equilibrium state has a corresponding artifact state combination.

In Figure 11 that details the "no outstanding issue" state, each state and sub-state has a corresponding artifact state combination:

- **No outstanding issue**

$$(\forall(i \in I)|(state(i) = resolved))$$

  - **Equilibrium:** all win conditions are with state "covered" showing that they are all reconciled and all issues are with state "resolved".

$$(\forall(w \in W)|(state(w) = covered)) \wedge (\forall(i \in I)|(state(i) = resolved))$$

  - **Enter win condition:** there is a new win condition whose state is "free" since it has been neither covered by an agreement nor involved in an issue; or there is a win condition dropped.

$$(\exists(w \in W)|(state(w) = free)) \vee$$
$$(\exists(w \in W)|(status(w) = inactive))$$

  - **Propose agreement:** there is an new agreement that is just proposed with state "open" to cover the new win condition

$$(\exists(a \in A)|(state(a) = open))$$

– **Vote for agreement:** a vote is conducted on the agreement to turn its state "vote-in-progress".

$$(\exists(a \in A)|(state(a) = in - progress))$$

Figure 15 demonstrates a subset of the consolidated model. For every state in the Equilibrium Model, it has a corresponding artifact life cycle combination. This life cycle combination reflects a collections of artifact connected by relationships (i.e. artifact chains). These artifact chains again represent a specific blending of inter-win-condition relationships. In the WinWin equilibrium state, all win conditions are reconciled and covered into some passed agreements. We discuss its corresponding artifact life cycle combination previously. To make sure that every single win condition is covered and every single issue is resolved, every artifact chain must contain a sub-chain of "Win Condition(covered) $\stackrel{involves}{\longrightarrow}$ Issue (resolved)". By definition of "covered" and "resolved", there must exist a "used" option that is adopted by a "passed" agreement. It can be proved that for all win conditions $W(H_i)_j$'s, the intersection of their win regions must be non-empty. "Resolve single issue" and "Resolve multiple issues" can be reasoned using the same framework as "Equilibrium state".

Equilibrium Model	Artifact Life Cycle	Inter-Artifact Relationship	Inter-Win Condition Relationship
equilibrium state	$\forall w \in W$ $state(w) = covered$ $\forall i \in I$ $state(i) = resolved$	Every artifact chain CH contains the following solid subchain: covered W I O A unused failed used passed resolved O A	$\bigcap_{\forall i} R_w(H_i)_j \neq \phi$ $\forall j$
Resolve single issue	$\exists!\ i \in I$ $state(i) \neq resolved$	There exists only one artifact chain CH, ≠covered ≠resolved W I	$\exists!\ I_k$
Resolve multiple issues	$\exists i_1, i_2 \in I, i_1 \neq i_2$ $state(i_1) \neq resolved$ $state(i_2) \neq resolved$	There exist artifact two chains CH_1 and CH_2 ≠covered ≠resolved W I CH_1 ≠covered ≠resolved W I CH_2	$\exists\ I_1, I_2$ s.t. $I_1 \neq I_2$

Figure 15: An integrated model of the many views

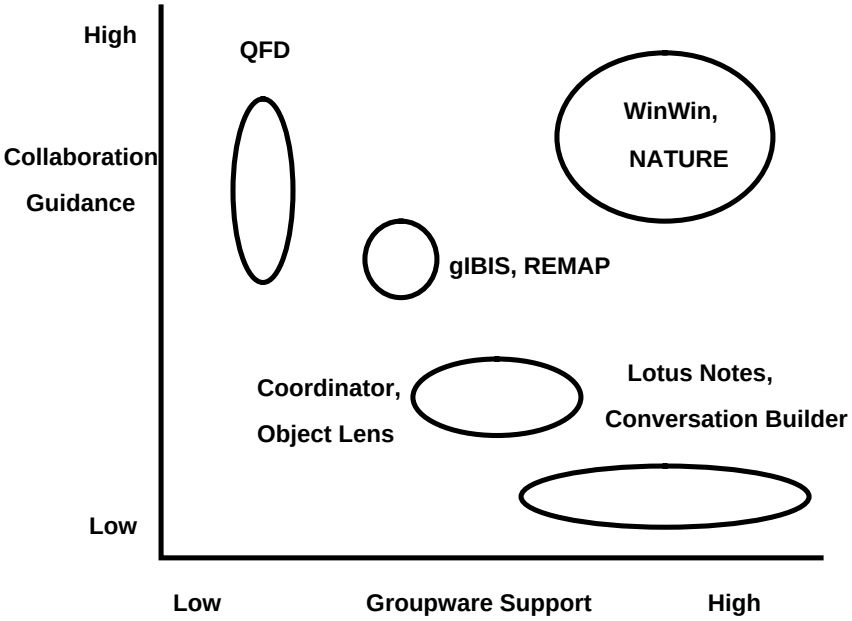

Figure 16: Needs versus capability comparison

5 Related Works

Figure 16 compares related works in terms of groupware support and collaboration guidance.

Among the groupware tools, Object Lens[LYM88] uses semi-structured objects combining object-oriented databases, hypertext, electronic messaging, and rule-based intelligent agents to provide a "spreadsheet" for cooperative work. The Coordinator[FGHW88] drives actions with a semi-structure message system. Lotus Notes[Mar91] provides shared document manager, replication algorithm, and customer contact tracking to facilitate a distributed Collaborative Editing System[GS87]. All these tools failed to provide a sound information structure where relations among system objectives should be established, examined and reasoned to reach a qualitative consensus. Even the most widely-used ones like the Coordinator and Lotus Notes offer only semi-structured messages with limited navigation support. As these tools are mainly technique-oriented, they are lack of systematic approach to address cooperation between cross functional teams as noted in [JK94].

For the approaches that set up collaborative goals, IBIS(Issue-Based Information System)[CY91] is a widely-adopted approach for establishing relations

among system objectives. gIBIS [CB88] is a graphical hypertext tool for setting up the semantic link according to the IBIS structure. Capturing planning dialogue for design rationale facilitates group decision-making. The Quality Function Deployment (QFD) approach[JK94] is also an IBIS-based system. It adapts "House of Quality (HoQ) Items" to localize possible conflicting requirements. One of the major problems with this approach is that focusing on customers brings potential conflicts when other stakeholders join the work later in the software life cycle. Another limitation lies on the difficulty of automating the original pencil-paper approach. A great percentage of negotiation is still done off-line. Although IBIS structures do avenue analysis, no tools are provided for analyzing trade-off so the design decision may overlook optimal solutions.

The Nature (Novel Approaches to Theories Underlying Requirements Engineering) project[P+94] is a joint effort among institutes in Europe. They aim at applying AI techniques to requirements engineering. Among works that have contributed to NATURE, Viewpoint[NKF94] is analogous to WinWin. Its focus is more on reconciling the heterogeneous representations of requirements than the contents (semantics of requirements). It does not provide a formal process model like the WinWin Spiral model or the elaborated WinWin Equilibrium to drive the reconciliation.

In summary, WinWin distinguishes from most traditional requirements engineering approaches in the following aspects:

1. Traditional requirements engineering approaches emphasize on user requirements. There is insufficient support to acquire requirements from stakeholders like customers and designers. Nor is there negotiation theory embedded to reconcile multiple views. WinWin, however, uses Theory-W to converge on system objectives and achieve view integration.

2. Traditional requirements are mostly rigid whereas win conditions in WinWin are negotiable. Design rationale is documented in the WinWin artifact to provide a corporate memory. Risks are explicitly addressed in WinWin to pinpoint possible breakdowns and propose early fixes. This makes it easy to increment and evolve requirements in the spiral model.

3. Past groupware tools do not connect with requirements engineering approaches very well because of the lack of scalable information structures for dealing with the requirements. The WinWin artifact structure can be easily exported to hypertext format and be viewed using web browsers with important information highlighted.

6 Conclusions

The research describes in this paper models the requirements negotiation infrastructure and dynamics. It identified some initial incompleteness and inconsistencies in the WinWin system and facilitates artifact change management. The

research involves formal and semi-formal descriptions of multiple views for the WinWin requirements negotiation system, including

- Win Condition Interaction: problem space view identifying the Win space of a stakeholder and how it intersects with other stakeholders' Win region,
- Artifacts and Relationships: schemata and entity-relationship model with functional definitions to support the negotiation infrastructure,
- Artifact Life-Cycles: state diagrams and predicate calculus to demonstrate how artifact evolve in the negotiation process
- System Equilibrium: hierarchical state model and predicate calculus to guide the WinWin users to recover the equilibrium (WinWin) state.

As stressed previously, potential problems coming with heterogeneous presentations for the same system are inconsistency and confusion. We present how we determine the relationship among the many views to achieve an integrated model. Section 2 shows how the WinWin Artifact relationships are determined as functions of various problem space views. Section 3 defines the Artifact Life Cycle Model as a function reflecting both the negotiation state indicated by the equilibrium model and the inter-artifact relationships. Therefore, it bridges the WinWin artifact relationships and the Equilibrium model. Section 4 demonstrates our approach of reconciling the many views. Future work will emphasize generalizing our reconciliation methodology and apply it to other multi-view frameworks.

References

[BBHL94a] B.W. Boehm, P. Bose, E. Horowitz, and M.J. Lee. "Experimental Results from a Prototype Next-Generation Process Support System". *TRW Systems Integration Group Technology Review*, 2(1):4–27, Summer 1994.

[BBHL94b] B.W. Boehm, P. Bose, E. Horowitz, and M.J. Lee. "Software Requirements As Negotiated Win Conditions". In *Proceedings First International Conference on Requirements Engineering*, pages 74–83. IEEE Computer Society Press, April 1994.

[BBHL95] B.W. Boehm, P. Bose, E. Horowitz, and M.J. Lee. "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach". In *Proceedings 17th International Conference on Software Engineering*, pages 243–253. ACM Press, April 1995.

[BR89] B.W. Boehm and R. Ross. "Theory-W Software Project Management: Principles And Examples". *IEEE Transactions on Software Engineering*, 15(7):902–916, July 1989.

[Buc94] R. Buchness. "The WinWin Spiral Model: Rockwell Just in Time Training Course". The Los Angeles Software Process Improvement Network (LA-SPIN) Meeting, November 1994.

[CB88] J. Conklin and M.L. Begeman. "gIBIS: A Hypertext Tool for Exploratory policy Discussion (graphical Issue Based Information Systems)". *ACM Transactions on Office Information Systems*, 6(4):303–331, October 1988.

[CY91] E.J. Conklin and K.C.B. Yakemovic. "A Process-Oriented Approach to Design Rationale". *Human-Computer Interaction*, 6:357–391, 1991.

[Dav90] A.M. Davis. "Specification In a World of Ever-Changing Requirements". In S. Andriole, editor, *Advanced Technologies for Command and Control systems Engineering*, pages 32–47. Fairfax, Va.: AFCEA International Press, 1990.

[EGR91] C.A. Ellis, S.J. Gibbs, and G.L. Rein. "Groupware: Some Issues and Experience". *ACM Communication*, 34(1):38–58, January 1991.

[FGHW88] F. Flores, M. Graves, B. Hartfield, and T. Winograd. "Computer Systems and the Design of Organizational Interaction". *ACM Transactions on Office Information Systems*, 6(2):153–172, April 1988.

[GS87] I. Greif and S. Sarin. "Data Sharing in Group Work". *ACM Transactions on Office Information Systems*, pages 187–191, April 1987.

[JK94] S. Jacobs and S. Kethers. "Improving Communication Decision Making within Quality Function Deployment". In *1st Intl. Conf. on Concurrent Engineering, Research and Application*, Pittsburgh, USA, August 1994.

[Lee95] M.J. Lee. "Formally Modeling the WinWin Requirements Negotiation Systems". In *Proceedings Doctoral Consortium of the Second IEEE International Symposium on Requirements Engineering*, March 1995.

[Lee96] M.J. Lee. "Foundations of The Winwin Requirements Negotiation System". Technical Report TR-96-501, USC Center for Software Engineering, University of Southern California, University Park, Los Angeles, April 1996.

[LYM88] K. Lai, K. Yu, and T.W. Malone. "Object Lens: A 'Spreadsheet' for Cooperative Work". *ACM Transactions on Office Information Systems*, 6(4):332–353, October 1988.

[Mar91] D.S. Marshak. "Lotus Notes, a Platform for Group Information Management Application". *Patricia Seybold's Office Computing Report*, June 1991.

[NKF94] B. Nuseibeh, J. Kramer, and A. Finkelstein. "A Framework for Expressing the Relationships between Multiple Views in Requirements Specification". *IEEE Transactions on Software Engineering*, 20:760–773, October 1994.

[P+94] K. Pohl et al. "Applying AI Techniques to Requirements Engineering: The NATURE Prototype ". In *Proceedings ICSE-Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence*, May 1994.

[T+96] K. Takahashi et al. "Hypermedia Support for Collaboration in Requirements Analysis". In *Proceedings Second International Conference on Requirements Engineering*, pages 31–40. IEEE Computer Society Press, April 1996.