Conceptual Design Model based Requirements Analysis in the WinWin Framework for Concurrent Requirements Engineering[1]

Prasanta Bose

Center for Software Engineering
Department of Computer Science
University of Southern California,

Los Angeles, CA 90089-0781

(Phone: 213-740-7275, bose@sunset.usc.edu)

Submitted to IWSSD-8

Submission Type: Technical Paper
Themes: requirements engineering, requirements analysis, software architecture modeling

Abstract

The WinWin framework provides a domain independent framework for the stakeholders to collaborate and negotiate in the requirements engineering phase of the software lifecycle. Requirements engineering in the framework leads to defining a win-win requirements model expressed using a set of conceptual elements that record stakeholder's objectives, constraints, concerns and negotiated agreements. A major problem confronted in the current WinWin framework is win-win requirements model analysis that lead to mapping the win-win requirements model which is primarily problem oriented to a solution-oriented requirements specification model that aids in win condition analysis and consequently negotiation.

This paper presents a constructive and goal-directed modeling approach to aid in win-win requirements model analysis. The approach involves concurrently elaborating a high-level conceptual design model along with the win-win model creation. The design model representation makes explicit partially specified constraints that form the conceptual architectural basis of the win-win requirements model and aids in win-win requirements model analysis. In this paper we present the key ideas of our approach: a) defining a domain-independent ontology for conceptual modeling of high-level design and its relationship with the WinWin model b) win-win artifact based representation of analysis goals and an abstract domain independent theory that encapsulate the conditions for satisfying the goals, and c) defining capabilities of an extended WinWin support system that to aid in analysis.

1.This research is sponsored by the Advanced Research Projects Agency (ARPA) through Rome Laboratory under contract F30602-94-C-0195 and by the Industrial Affiliates of the Center for Software Engineering.

1 Introduction

In today's rapidly evolving technology and competitive marketplace - it has become critical that the design of complex software involve the users, designers, maintainers, and other stakeholders concurrently in the initial stages of design when requirements and system architectural decisions are made, since these decisions have significant impacts on product quality, cost, and schedule. The challenge is providing collaboration support that aid in such concurrent decision making.

Two major problems arise in developing computer supported collaborative approaches to concurrent requirements engineering:

i) Decision rationale capture and decision coordination involving the different stakeholder activities.

ii) Requirements analysis that facilitates requirements negotiation and leads to a specification model.

The WinWin framework, being developed and experimented at the USC Center for Software Engineering, provides a solution to the first problem. The framework makes use of semistructured representations of artifacts called win conditions, issues, options and agreements to represent stakeholder objectives and constraints, conflicts, conflict resolving strategies and their agreements respectively. The WinWin process leads to defining a win-win requirements model that is expressed using the WinWin artifacts and relations between them. The requirements model explored and elaborated in the WinWin framework in essence captures stakeholder-oriented objectives, options and constraints in the form of a decision rationale.

A major problem confronted in such a concurrent requirements engineering framework is win-win requirements analysis for understanding WinWin artifact interactions and negotiating requirements. This paper presents a constructive and goal-directed modeling approach - in which a partial conceptual design model gets elaborated concurrently with the win-win model creation guided by the WinWin process goals to obtain agreements on issue-free functional objectives and non-functional objectives defined by the winconditions. In such an approach the WinWin artifacts get mapped[2] to and related to the conceptual design model elements. The constructed model and its relations to winwin model provide a conceptual architectural basis of the win-win requirements model and thereby facilitate causal understanding of how win conditions may be achieved by design elements and the ramifications of the design on the win conditions. The key ideas involve: a) defining a domain-independent ontology for conceptual modeling of high-level design and its relationship with the WinWin model b) win-win artifact based representation of analysis goals and an abstract domain independent theory that encapsulate the conditions for satisfying the goals - the abstract theory in essence defines a declarative specification of an extended WinWin process model for win-win requirements model analysis via

2. We are using the term mapping in a very restricted sense. It is used here is to denote the set of semantic links relating the win-win requirements model elements and the design model elements.

partial design model construction and c) defining capabilities of an extended WinWin support system that aid in analysis.

1.1 Outline

In the next section we describe the WinWin approach - the winwin process model, the ontology for stakeholder oriented requirements decisions, and support framework. Section 3 provides the ontology for high-level design models, its relationship with the winwin requirements model and defines a declarative theory of WinWin states that define analysis goals. The theory aids in defining the WinWin process and provide a systematic basis for goal-directed winwin requirements analysis. Section 4 and 5 presents related work and summary of this paper respectively.

2 The WinWin Approach

The WinWin approach [Boehm95] is aimed at addressing collaboration for the requirements engineering phase of the software life-cycle. The three key ideas in the approach are:

- **WinWin Spiral process.** The process [Boehm94] consists of two main steps: i) Identifying stakeholders and their win conditions ii) Creating win-win relationships collaboratively and negotiating to confront win-lose and lose-lose situations. A key aspect of the process is that it introduces economic, product quality and risk considerations into the decision making steps and introduces tradeoff exploration into the process to address risks and conflicts. The process identifies the domain independent conceptual elements that forms the agreed upon ontology of decisions for collaboration and negotiation and also identifies the domain taxonomy, as a given, for providing a shared understanding of the domain. The taxonomy is used as a domain-specific reference model with respect to which stakeholders express their win conditions, issues and agreements.
- **Win-Win Requirements model.** The decision space that gets collaboratively explored in a WinWin process is modeled using four main conceptual objects: i) Win Condition - capturing the desired objective of the individual. ii) Issue - capturing the conflict between win conditions and their associated risks and uncertainties. iii) Option - capturing a strategy for resolving an issue. iii) Agreement - capturing the agreed upon set of conditions which satisfy stakeholder win conditions and also define the system objectives. The ontology also defines a set of relations between these objects. Figure 1, shows a typical abstract structure of the decision rationale in terms of the above entities and the link types denoting the relations between them. As shown in the figure, an issue (I) is related to one or more win conditions (W_x and W_y) through the *involves* relation. An option (O) for resolving an issue (I) is related to the issue through the *addresses* relation. An agreement (Ag_i) based on an option choice (Op) is related to Op through the *adopts* relation. The agreement (Ag_i) has a *covers* relation with a win condition and a *replaces* relation to any previous agreement it substitutes. Table 1 provides a schema-based representation of each of the artifacts. Each object type is defined by a set of

tuples, where each tuple consists of the slot name denoting the relation name with a single or multiple cardinality and a typed value field that ranges over an object or enumerated value set. For example, in the win condition object W, the relation comment is a one-to-many function between win conditions and the set of all strings. Similarly the relation defined by the adopted-by slot in an option is a one-to-one function between the options and agreements. The state slot of each object is a one-to-many function from the object to an enumerated set of unary predicate constants. For example, a win condition can be in both active and at_issue state. A declarative theory[3] is used to define states of objects and their dependence on states of other related objects. The win-win requirements model that gets constructed by the WinWin process is based on domain-specific instances of the above WinWin meta objects and their relations.

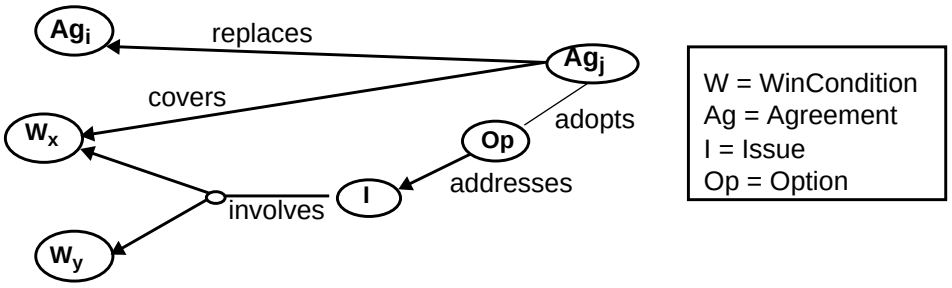

FIGURE 1. The WinWin decision objects and relations between them.

- **WinWin support framework**. Figure 2 shows a schematic diagram of the support framework that implements the WinWin concept of collaboration. As shown in the figure, each stakeholder is associated with a WinWin client. The clients communicate

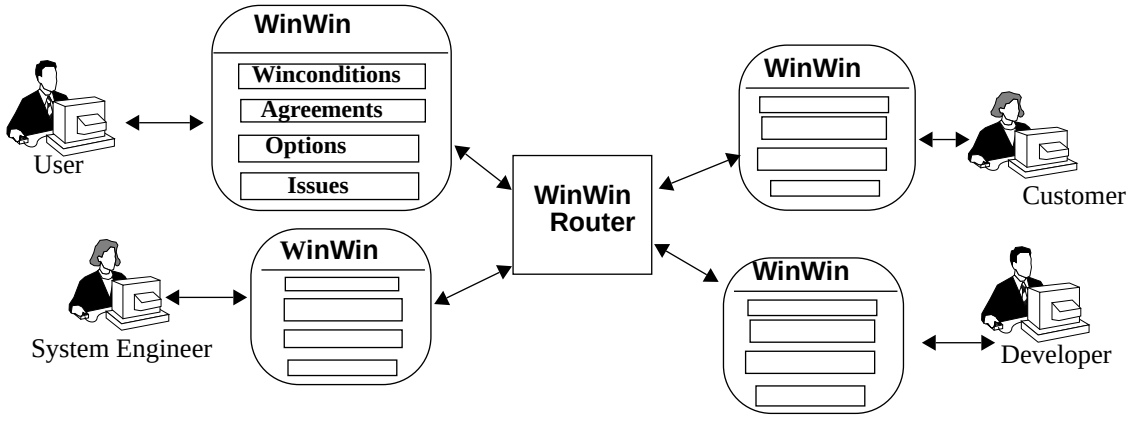

FIGURE 2. A Schematic diagram of the WinWin architecture.

via the WinWin Router. A stakeholder interacts with the WinWin client support system interface to define their individual win conditions, raise issues, suggest options and draft and vote on agreements. The WinWin objects created or revised by the

3. The abstract theory, defining the states and their dependence, and its use in guiding coordination for incremental construction of the WinWin requirements model is described in [Bose 95].

a) Wincondition(W):

Slot name	Cardinality	Value-Type
<name>	single	string
<Objective>:	single	strings
<comment>:	multiple	string
<sub-winc>:	multiple	win condition
<state>:	multiple	predicate
<rationale>:	multiple	reason
<covered-by>:	single	agreement
<involved-in>	single	issue
<replaces>	single	wincondition

State predicates: {active, covered, at-issue, retracted, valid, reduced}

b) Option(O)

Slot name	Cardinality	Value-Type
<name>	single	string
<body>:	single	string
<adopted-by>:	single	agreement
<assertion>:	multiple	constraint
<pros>:	single	string
<cons>:	single	string
<rationale>:	multiple	reason
<rejection>:	multiple	reason
<involved-in>:	multiple	issue
<state>:	multiple	predicate

State predicates: {admissible, valid, adopted, at_issue, rejected, retracted}

c) Issue(I):

Slot name	Cardinality	Value-Type
<name>	single	string
<body>:	single	string
<involves>:	multiple	{W, O, A}
<rationale>:	multiple	reason
<addressed-by>:	multiple	option
<state>:	multiple	predicate

State predicates: {resolved, unresolved, non-resolvable, addressed, retracted}

d) Agreement(A)

Slot name	Cardinality	Value-Type
<name>:	single	string
<body>:	single	string
<commitments>:	multiple	constraint
<covers>:	multiple	{W, A}
<involved-in>:	multiple	issue
<adopts>:	single	option
<state>:	multiple	predicate
<replaces:>	single	agreement

State predicates: {open, valid, committed, at_issue, replaced, retracted}

TABLE 1. A subset of the WinWin ontology for Decision rationale.

stakeholder are recorded in a local database by the WinWin client. Any update operation performed by a stakeholder are noted by the client and used to notify other clients and update their object base using the WinWin Router. Hence each client, in essence keeps a copy of all the WinWin objects (winconditions, issues, etc.) - all of which may not be modifiable by the associated stakeholder. The issues raised are used to focus negotiation between stakeholders. WinWin also provide tools to support the negotiation activity [Boehm95]. Issue resolution leads to negotiated changes in win conditions. When all issues have been resolved, agreements are drafted and closed.

3 Win-Win Requirements Model Analysis

Given the above WinWin framework for collaboration, the elicitated win-win requirements model represents stakeholder oriented needs and constraints. Analyzing the win-win requirements model for understanding issues and guiding negotiation of tradeoffs requires a solution-oriented model at a level of abstraction relevant to understanding the win-win artifact interactions. Figure 3 shows an initial conceptualization of the extended

decision rationale structure that builds on the viewpoint that the process of analysis is best facilitated by mapping the requirements model to an abstract design model. The extensions in the decision rationale structure involve use of a design model (D) that is motivated by the win conditions (W), considering design factors (Df) arising from D for explaining issues and introducing revisions (del-D and del-W) based on options.

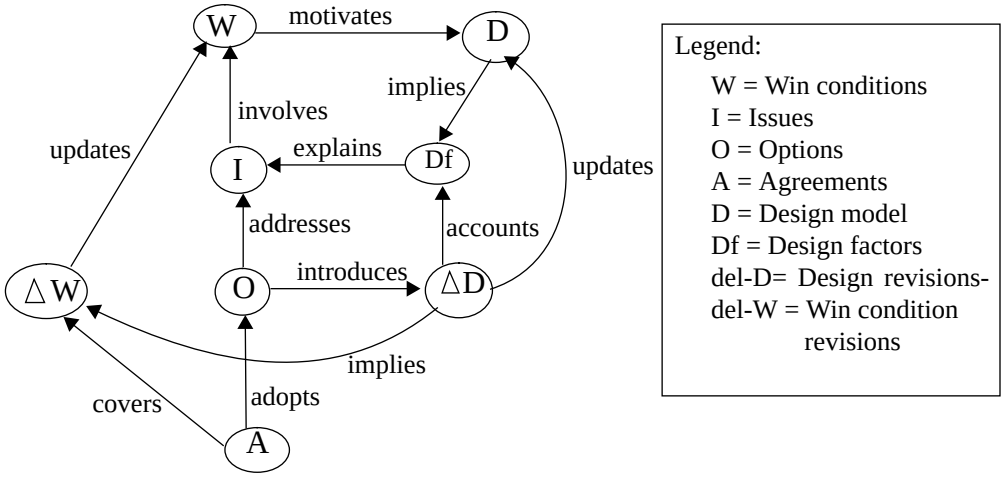

FIGURE 3. An initial conceptualization of the decision structure supporting analysis of win-win requirements model.

We can further refine the above conceptualization of the abstract design model and its relationship to the requirements model based on a more detailed understanding of the needs that must be met by the design representation:

i) Win conditions expressing functional objectives need to be mapped to functional-elements and their task assignment - such a representation, explicating the active information processing design elements and their task responsibility then serves as the basis for attributing constraints on design features of the function elements in order to achieve non-functional objectives.

ii) Win conditions expressing non-functional objectives (primarily system quality specific properties) need to be situated for relevant tasks and function elements and mapped to constraints on those elements - the constraints make explicit behavioral, communication, structural, and other architecture oriented characteristic features of function-elements that strengthens quality properties

iii) Issues expressing win condition conflicts have their basis in design factors - the design factors make explicit the negative ramifications of existing design feature constraints on functional and non-functional needs.

iv) Options expressing strategies (solution specific ones only) for resolving issues need to be mapped to revision/tradeoff constraints - the revision constraints make explicit the necessary changes required to resolve issues pertaining interfering ramifications of design choices. Such revisions may impact the win-win requirements model and/or the solution model being constructed.

The point to note is that the abstraction being used in the above mapping of the win-win requirements model must be adequate to capture multiple views (for example information processing view, structural-view, communication view and etc.). Such representational

adequacy is required to map and understand different stakeholder win conditions that bear on different aspects of the underlying design model elements and their relationships. The example described in the next subsection briefly illustrate the use of the above concepts in doing win-win requirements model analysis.

3.1 An Example

As shown in the scenario -- the unshaded portion of figure (a) - (d) illustrate the incremental construction of the win-win requirements model. The corresponding shaded portion of figure show the design model that gets concurrently elaborated with the win-win model. In Figure 4(a), the functional objective (O_f) on visualizing alarms in real-time, of the wincondition W_1 is supported by the information processing task (T) of displaying sensor signals in real-time, where T is assigned to a signal-processing function element (FE). Figure 3(b) shows the result of addition of a new wincondition W_2 which imposes the non-functional constraint (O_q) of portability of the software. The non-functional constraint (O_q) is met by introducing the design constraint (C_1) of infrastructure independent resource bindings that applies-to FE. The design constraint (C_1) raises the design factor (Df) that infrastructure dependence determines real-time performance properties (figure 4(c)). The Df is then used to explain the issue (I) on real-time performance of the FE. The issue is addressed by the option O which defines a strategy (S) of using portability layers (figure 4(d)). The strategy S is mapped to the constraint C_2 of platform-independent interface element that applies to the FE. Accepting this option leads to an agreement A which settles the issue I.

3.2 Conceptual Design Ontology

Based on the insights developed at the beginning of this section, we minimally extend the WinWin ontology with the following additional set of domain-independent objects (as part of the meta-model) that allow constructing a domain specific high-level design model in conjunction with the win-win requirements model:

- *Objective*: represents problem domain specific functional and non-functional or quality objectives and desired process constraints (e.g. schedule and cost)
- *Function-element*: represents domain specific active entities that participate in achieving tasks (e.g. humans, software, hardware - our focus is primarily on software-based systems and humans). Function-elements are capable of communicating with other function-elements, manipulating information elements and controlling behavior of other function-elements. With regard to software-based systems, function-elements are abstractions of implemented modules and components. Function-elements are characterized by architecture oriented features representing information communication, structural, interaction behavior, control and other aspects.
- *Information-element:* represents domain-specific information that gets consumed by function-elements to perform tasks or gets produced as a result of performing tasks.

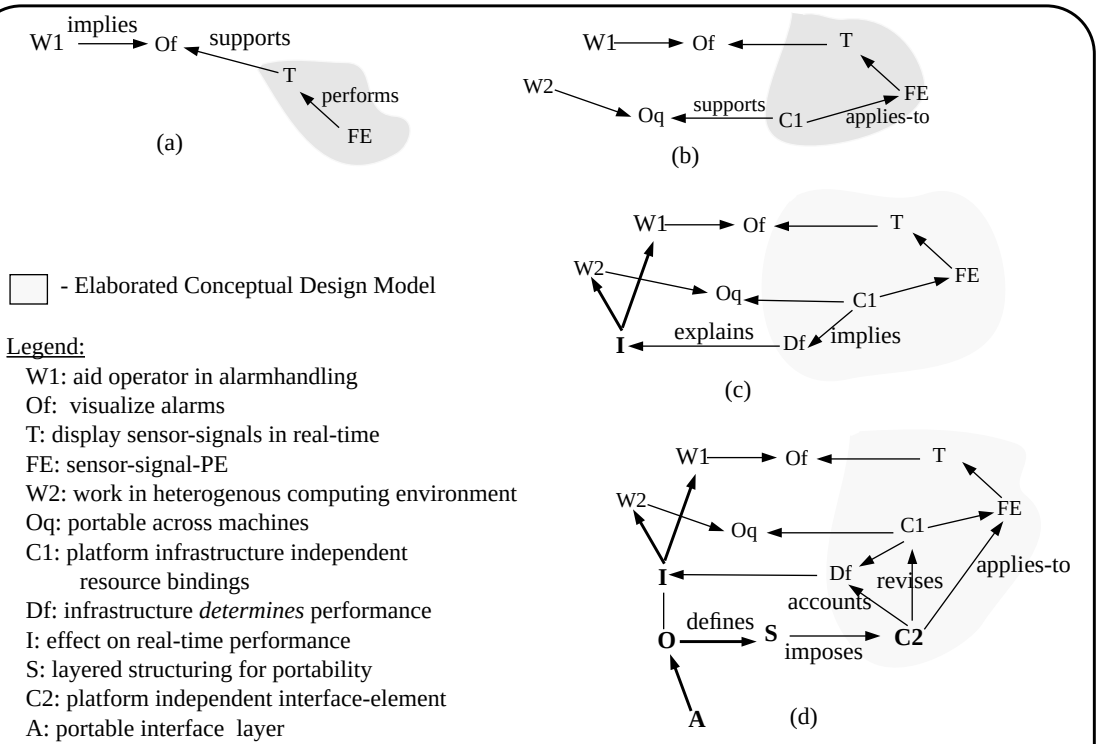

FIGURE 4. An example ccenario showing the mapping of the win-win requirements model to a high-level design model.

• *Task*: represents domain specific information manipulation activities that are assigned to function-elements to achieve functional-objectives

• *Constraint:* has subtypes that represent task specific and function-element specific feature constraints. The function element specific feature constraints correspond to constraints on function-element information communication, function-element-behavior, function-element-control-coupling, and information-element produced and consumed features[4].

• *Design-factor:* represents the design feature and its ramification on an objective.

• *Design-strategy*: represents design feature constraints necessary to support an objective. Design-strategies represent pieces of domain-specific design knowledge.

The constraint object plays an important role in the design model construction. Design-element types, and their features specified by the constraints lead to defining the solution architecture[5]. The body of the constraint object[6] is represented using constraint assertions. Relations are used in specifying object attributes and link types between objects. The important relations in the design model are:

4. We are in the process of defining a more complete taxonomy of design features and their value constraints.

- a task is *assigned_to* a function-element
- a constraint is *applied_to* one or more function-elements, one or more tasks
- a constraint *implies* a design-factor
- a design-factor *interferes* with an objective
- a function-element *consumes/produces* a function-element

The important relations linking WinWin model objects to design model objects are:

- a functional objective is *supported-by* a task and a task *supports* a functional-objective
- a non-functional objective *motivates* a constraint and a constraint *supports* a non-functional objective
- a non-functional objective is *relevant-to* one or more tasks or function-elements
- an issue is *explained-by* a design-factor and a design-factor *explains* an issue
- a design-strategy *imposes* constraints

3.3 Mapping Win-Win Artifacts to Design Elements: A Minimal Theory

Given the above ontology of the design model - the problem then is instantiating those objects to construct the abstract design model and linking the instantiated elements to win-win requirement model elements. Since the win-win requirements model elements are represented in semistructured terms - domain-specific and automated approaches to mapping is very difficult. We adopt a semi-automated approach to this problem. We define an abstract and minimal theory that articulates the goals of analysis and specify the criteria for their satisfaction in terms of conditions for existence of object types and their relationships in the two models. The user can then instantiate the theory to construct and elaborate a partial design model and link the model to the existing win-win requirements model based on his/her application domain knowledge.

Based on the understanding developed so far on the analysis needs, the conceptual design model to support the needs, we can articulate the win-win requirements model analysis as: (i) obtaining function-elements and their assigned tasks for supporting functional-objectives (ii) obtaining function-element feature constraints for realizing quality-objectives (iii) obtaining ramifications of design features on quality objectives (iv) obtaining revisions and tradeoffs. We represent these goals as desired states of win-win artifacts and define a domain independent declarative theory that articulates the conditions for satisfying those states. The specified conditions when satisfied lead to elaborating the design model and defining its relationship to the win-win requirements model. Table 2 describes such a theory.

5.It is to be noted that the concept of style used in some of the recent research work on software architectures [PERR92, ABOW93, MORI94] can be abstractly represented by one or more function-elements and a set of constraints on them.

6. We do not discuss any further the underlying schema-based representation of the objects.

The portion of the theory in Table 2 (i) define conditions for wincondition states that ensure that (a) the win condition involving functional objective gets mapped (represented by the state predicate t_mapped) to a information processing task, (b) the task relevant to the win condition has been assigned to a function-element, (c) the win condition involving quality objectives get situated relative to one or more function-elements and/or tasks and (d) the quality objective gets elaborated by defining constraints on features of function-elements and/or tasks that are deemed relevant to the quality objective. The theory specified in Table 2(ii) leads to defining issues based on design-factors that arise in the context of constraints interfering with objectives. The theory defined in Table 2(iii) leads to revisions based on constraints imposed by design strategies.

TABLE 2. A subset of states of WinWin objects for representing analysis goals and the theory defining the conditions for satisfaction of those states expressed in terms of the design model.

i) Win Condition(I): states = { *t_mapped, t_assigned, q_situated, q_elaborated}*

t_mapped(W) $\Leftrightarrow$ $\exists$ F:objective, functional_objective(W, F) $\rightarrow$ $\exists$ T_i:task, i>=1, supports(T_i,F)

t_assigned(W) $\Leftrightarrow$ $\exists$F:objective, functional_objective(W, F)
Λ supports(T,F) $\rightarrow$ $\exists$ Fe:function_element, assigned_to(T, Fe)

q_situated(W) $\Leftrightarrow$ $\exists$ Q:objective, quality_objective(W, Q) $\rightarrow$
[$\exists$ T:task, relevant_to(Q,T)] V [$\exists$ Fe:function_element, relevant_to(Q, Fe)]

q_elaborated(W) $\Leftrightarrow$ $\exists$ Q:objective, quality_objective(W, Q)
Λ [[relevant_to(Q,T) $\rightarrow$ $\exists$ C:constraint, motivates(Q, C) Λ applies_to(C, T)]
V [relevant_to(Q, Fe) $\rightarrow$ $\exists$ C:constraint, motivates(Q, C) Λ applies_to(C, Fe)]

ii) Issue(I): states = { *explained }*

explained(I) $\Leftrightarrow$ $\exists$ Df: design_factor, explains(DF, I) $\rightarrow$
[$\exists$ Q: objective, C: constraint, implies(C, DF) Λ interferes(DF,Q)]
V [$\exists$ Q: objective, interferes(DF, Q)]

iii) **Option(O): states = *{motivated, applied}***

motivated(O) $\Leftrightarrow$ $\exists$ S: design_strategy, defines(O, S) $\rightarrow$
[$\exists$ Df: design_factor, C: constraint, imposes(S, C) Λ accounts(C,Df)]

applied(O) $\Leftrightarrow$ $\exists$ S: design_strategy, defines(O, S) Λ $\exists$ C: constraint, imposes(S, C) $\rightarrow$
[$\exists$ C': constraint, revises(C, C')] V [$\exists$ O: objective, constraints(C, O)]

3.4 Supporting Analysis in WinWin Support System

The current implementation of the WinWin support system [Boehm95] does not provide any support capabilities that facilitate construction of the design model and its linking to the win-win requirement model. We are experimenting with additional capabilities based on the extended framework developed in the earlier subsections. The capabilities involve:

i) Providing templates for the new design objects that get instantiated by the user ii) Providing analysis process support by generating analysis goals for users of WinWin based on win-win artifact states and generating partial analysis plans based on the abstract theory to support achieving those goals.

4 Related Work

Our work on using a conceptual design model for win-win requirements model analysis is related to the work in the areas of software architectures, requirements elicitation, viewpoint modeling, and design-rationale. The conceptual design ontology that we have used is related to the high-level design modeling ontology being developed in the software architecture area [PERR92, ABOW93,GARL93, MORI94]. We have attempted to define an ontology at meta-modeling level that can then be instantiated and refined in a least committed manner. This is similar in spirit to the work of [DARD91] - who also use a meta-modeling approach to requirements elicitation and analysis. The major difference with their work is that our metamodel is constrained by the nature of analysis that the winwin requirements model calls for.

Our work is also related to the work in the area of decision rationale modeling. The conceptualization used in defining the relations between elements in the two models is in the same spirit of the work of [GRUB92, LEE90,CUTO93,CONK88]. The other relevant research is that of use of viewpoint models [NUSE93] in acquiring and analysis of requirements specifications. The viewpoints work makes use of multiple representations for capturing different stakeholder perspectives and the analysis is based on inter-viewpoint analysis rules. In our work we have attempted to define a uniform architecture oriented representation that can be use to map the stakeholder specific win conditions.

5 Summary and Future Work

This paper presents ongoing research on analysis of the win-win requirements model that gets collaboratively generated in the WinWin framework for requirements engineering. The major ideas presented to address the analysis problem involve: i) defining a design conceptualization that aids to construct a solution oriented model for the win-win requirements model that is primarily stakeholder (as well as problem) oriented. and defining its relationship to the win-win model. ii) Defining an abstract theory for guiding constructing of the model and defining extended support capabilities based on the theory.

Our future work is aimed at: i) developing a more complete design model ii) defining a taxonomy of constraints on features (domain independent and domain independent) of the design model elements. iii) developing more structured representations of the design model elements and domain specific theories based on those representations to facilitate automated support capabilities for identifying issues. iv) Extending the existing WinWin Support system with the support capabilities identified and experimenting to understand its application in architecture-based requirements engineering.

Acknowledgments. The author gratefully acknowledges the helpful comments of Dr. Barry Boehm and support of the WinWin team at the Center for Software Engineering for implementation of the WinWin system.

6 References

1. [ABOW93] G. Abowd, R. Allen, D. Garlan. "Using Style to Understand Descriptions of Software Architecture", Proceedings of the First ACM SIGSOFT Symposium on the Foundations of Software Engineering, Software Engineering Notes, ACM Press, December 1993
2. [BOEHM88]. B.W. Boehm, "A Spiral Model of Software Development and Enhancement," *Compute*r, May 1988, pp. 61-72.
3. [BOEHM1994]. B. W. Boehm, P. Bose, E. Horowitz, and M-J. Lee, "Software Requirements As Negotiated Win Conditions", *Proceedings, 1994 International Conference on Requirements Engineering*, IEEE, April 1994.
4. [BOEHM95]. B. Boehm, P. Bose, Ellis Horowitz and Ming June Lee "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach", *IEEE Proceedings of the 17th ICSE Conference*, 1995.
5. [BOSE95]. P. Bose "A Model for Decision Maintenance in the WinWin Collaboration Framework", KBSE Proceedings, 1995 (to appear)
6. [CUTO93], Cutosky, Engelmore, Gruber, Genesereth, Mark, Tenenbaum and Weber, "PACT: An Experiment in integrating concurrent engineering systems", *Computer*, Vol. 26, No. 1, 1993.
7. [CONK88]. J. Conklin and M. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Trans. Office Info. Systems*, Oct. 1988, pp. 303-331.
8. [DARD91]. A. Dardenne, S. Fickas and A. Lamsweerde, "Goal-directed Concept Acquisition in Requirements Analysis", Proceedings of the 6th IWSSD, 1991.
9. [Curtis et. al. 1988]. B. Curtis, H. Krasner, N. Iscoe, "A Field Study of the Software Design Process for Large Systems", *CACM*, Vol. 31, No. 11, November 1988, pp 1268-1287.
10. [GARL93] D. Garlan and M. Shaw. "An Introduction to Software Architecture" Advances in Software Engineering and Knowledge Engineering, World Scientific Publishing Co., 1993
11. [GARL94] D. Garlan, R. Allen, and J. Ockerbloom. "Exploiting Style in Architectural Design Environments", Proceedings of the Second ACM SIGSOFT Symposium on Foundations of Software Engineering, Software Engineering Notes, ACM Press, December 1994
12. [GRUB92] T. R. Gruber amd D. M. Russell, "Generative Design Rationale: Beyond the record and replay paradigm", in *Design Rationale*, Lawrence Erlbaum Associates, 1992.
13. [GUIN88] R. Guindon and B. Curtis, "Control of Cognitive Processes during Software Design: What tools are needed?" *CHI* 1988.
14. [LEE90]. J. Lee, "SIBYL: A Qualitative Decision Management System," in *Artificial Intelligence at MIT: Expanding Frontiers*, P. Winston and S. Shellard, eds., MIT Press, 1990, pp. 106-133.
15. [NUSE93]. B. Nuseibeh, J. Kramer, A. W. Finkelstein, "Expressing the Relationship Between Multiple Views in Requirements Specification", Proceedings of the15th International Conference on Software Engineering, 1993.
16. [MORI94] M. Moriconi and X. Qian. "Correctness and Composition of Software Architectures", Proceedings of the Second ACM SIGSOFT Symposium on Foundations of Software Engineering, Software Engineering Notes, ACM Press, December 1994
17. [PERR92] Perry D. and Wolf A. "Foundations for the Study of Software Architecture". ACM SIGSOFT Software Engineering Notes, Vol. 17, 4, October 1992

18. [Ramesh-Dhar, 1992]. B. Ramesh and V. Dhar, “Supporting Systems Development by Capturing Deliberations During Requirements Engineering,” *IEEE Trans. SW Engr.*, June 1992, pp. 498-510.

19. [SHAW94] Shaw M., DeLine R., Klein D., Ross T., Young D., and Zelesnik G. Abstractions for Software Architecture and Tools to Support Them. Carnegie Mellon University, Pittsburgh, February 1994