8 The WinWin Approach: Using a Requirements Negotiation Tool for Rationale Capture and Use

B. Boehm, H. Kitapci

Abstract: A highly cost-effective approach for rationale capture and management is to provide automated support, and capture the resulting artifacts of the process by which software and system requirements and solutions are negotiated. The WinWin process model, equilibrium model, and collaborative negotiation tool provide capabilities for capturing the artifacts. The MBASE software process model provides an approach for using and updating the rationale artifacts and process to keep it in a win-win state. Supporting requirements negotiation with attaching rationale can have a high impact on all phases of development by enabling much better context for change impact analysis as the increasingly frequent requirements changes arrive. The WinWin approach involves having a system's success-critical stakeholders participate in a negotiation process so they can converge on a mutually satisfactory or win-win set of requirements. The WinWin framework in essence captures stakeholder-oriented objectives, options and constraints in the form of a decision rationale.

Keywords: requirements negotiation; WinWin negotiation approach; rationale capture; Theory W; WinWin spiral model, EasyWinWin

8.1 Introduction

Negotiation techniques are critical success factor in improving the outcome of software projects. At the University of Southern California's Center for Software Engineering (USC-CSE), we have been developing a negotiation-based approach to software and system requirements engineering, architecture, development, and management. Our approach has three primary elements:

1. *Theory W*, a management theory and approach, which says that making winners of the system's key stakeholders is a necessary and sufficient condition for project success [2].
2. *The WinWin Spiral Model*, which extends the spiral software development model [1] by adding Theory W activities to the front of each cycle.
3. *EasyWinWin*, a collaborative groupware negotiation tool that makes it easier for distributed stakeholders to negotiate mutually satisfactory (win–win) system specifications.

Defining requirements is a complex and difficult process, and defects in the process often lead to costly project failures [16]. Requirements emerge in a highly collaborative, interactive, and interdisciplinary negotiation process that involves heterogeneous stakeholders. The EasyWinWin approach involves having a system's success-critical stakeholders participate in a negotiation process so they can converge on a mutually satisfactory (win–win) set of requirements.

Some difficulties within requirements engineering, e.g., determining a feasible and mutually satisfactory set of requirements, are eliminated by achieving a reconciliation of customer expectations with developer capabilities before firmly committing to a set of requirements. A hard to achieve customer's or user's win condition will conflict with the developer's win condition to minimize the risk of delivering an acceptable product within budget and schedule. Conflicting requirements must be identified and negotiated, relevant alternatives must be made explicit and it must be assured that the "right" decision is made. In the WinWin approach, this conflict is identified as an issue needing resolution before stakeholders commit on the agreements.

The overall WinWin negotiation approach is similar to other team approaches for software and system definition such as gIBIS [9], SIBYL [13], and REMAP [15]. Our primary distinguishing characteristic is the use of the stakeholder win–win relationship as the success criterion and organizing principle for the software and system definition process. Our negotiation guidelines are based on the Harvard Negotiation Project's techniques [11].

In this chapter, we first introduce the WinWin Spiral Model. Next, we identify the fundamental concepts of WinWin model and the use of WinWin model in software development process. Then we introduce the EasyWinWin tool for converging stakeholders' interests to win–win agreements and the WinWin equilibrium state to test whether the negotiation process has converged. We provide an example of WinWin requirements negotiation results from our USC CS577 Software Engineering course projects. We then discuss how such results can serve as captured rationale for later user in avoiding mistakes in subsequent project decisions. We conclude with a discussion of using captured rationale to improve later decisions, related work, and future directions in requirements negotiation and rationale capture.

8.2 The Theory W and WinWin Spiral Model in Software Development Process

8.2.1 Theory W

The foundation for the WinWin approach is Theory W, a management theory similar to Theories X, Y, Z. Theory W's fundamental principle is that a necessary and sufficient condition for a successful enterprise is that the enterprise makes winners of all its success-critical stakeholders. It is well-matched to the problems of software project management. It holds that software project managers will be fully successful if and only if they make winners of all the other participants in the software process: superiors, subordinates, customers, users, maintainers, etc. This principle is particularly relevant in the software field, which is a highly people-intensive area whose products are often unfamiliar with user and management concerns.

Making everyone a winner may seem like an unachievable objective. Most situations tend to be zero–sum, win–lose situations. Nevertheless, win–win situations exist, and often they can be created by careful attention to people's interests and expectations. The best work on creating them has been done in the field of negotiation. The book "*Getting to Yes"* [11] is a classic in the area. Its primary thesis is that successful negotiations are not achieved by haggling from preset negotiation positions, but by following a four-step approach whose goal is basically to create a win–win situation for the negotiating parties (1) separate the people from the problem, (2) focus on interests, not positions, (3) invent options for mutual gain, (4) insist on using objective criteria.

The Theory W approach to software project management expands on these four steps to establish a set of win–win preconditions, and some further conditions for structuring the software process and the resulting software product.

8.2.2 WinWin Spiral Model

The original spiral model [1] uses a cyclic approach to develop increasingly detailed elaborations of a software system's definition, culminating in incremental releases of the system's operational capability. Each cycle involves four main activities:

- Elaborate the system or subsystem's product and process objectives, constraints, and alternatives
- Evaluate the alternatives with respect to the objectives and constraints. Identify and resolve major sources of product and process risk

- Elaborate the definition of the product and process
- Plan the next cycle, and update the life-cycle plan, including partition of the system into subsystems to be addressed in parallel cycles. This can include a plan to terminate the project if it is too risky or infeasible. Secure the management's commitment to proceed as planned

Since its creation, the spiral model has been extensively elaborated and successfully applied in numerous projects. However, some common difficulties led USC-CSE and its affiliate organizations to extend the model to the WinWin spiral model described in the following text.

WinWin Extensions: Negotiation Front End

One difficulty was determining where the elaborated objectives, constraints, and alternatives come from. The WinWin spiral model resolves this by adding three activities to the front of each spiral cycle, as Fig. 8.1 shows:

- Identify the system or subsystem's key stakeholders
- Identify the stakeholders' win conditions for the system or subsystem
- Negotiate win–win reconciliations of the stakeholders' win conditions

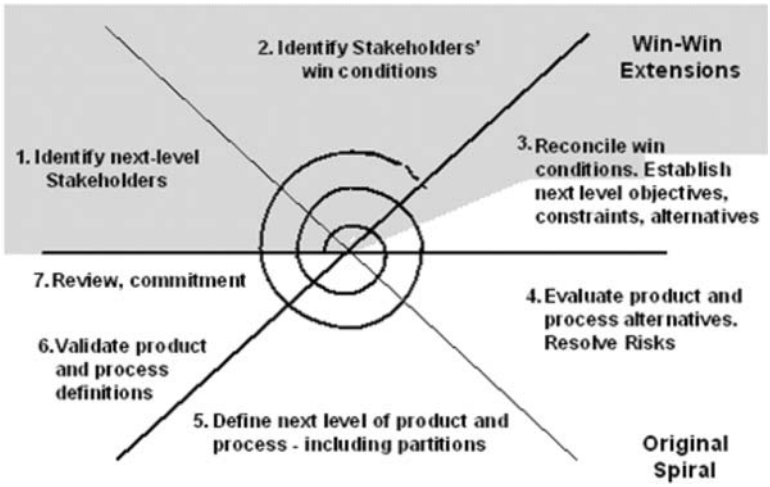

Fig. 8.1. The WinWin spiral model of software engineering includes front-end activities (*gray*) that show where objectives, constraints, and alternatives come from. This lets users more clearly identify the rationale involved in negotiating win conditions for the product

The new model adds front-end activities that show where objectives, constraints and alternatives come from. This lets stakeholders more clearly identify the rationale involved in negotiating win conditions for the product. A key aspect of the model 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.

Process Anchor Points

Another difficulty in applying the spiral model across an organization's various projects was that the organization has no common reference points for organizing its management procedures, cost and schedule estimates, and so on. This is because the cycles are risk driven, and each project has different risks. In attempting to work out this difficulty with USC-CSE's industry and government affiliates using our COCOMO II cost model [7], we found a set of three process milestones, or anchor points, which we could relate to both the completion of spiral cycles and to the organization's major decision milestones.

Over the years of developing electronic services applications for the USC Libraries, we have been evolving Model-Based Architecting and System/Software Engineering (MBASE). MBASE involves early reconciliation of a project's success models (correctness, business case, stakeholder winwin,…); product models (domain, requirements, architecture,…); process models (waterfall, evolutionary, spiral,…); and property models (performance, reliability,…). It extends the previous spiral model in two ways:

- Initiating each spiral cycle with a stakeholder win–win stage to determine a mutually satisfactory set of objectives, constraints, and alternatives for the system's next elaboration during the cycle.
- Orienting the spiral cycles to synchronize with a set of life cycle anchor points: Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC)

The LCO version focuses on establishing a sound business case for the package. It need only show that there is at least one feasible architecture. The LCA version commits to a single choice of architecture and elaborates it to the point of covering all major sources of risk in the system's life cycle. The LCA is the most critical milestone in the software's life cycle. The IOC version focuses on a workable initial operational capability for the

project including system preparation, training, use, and evolution support for user, administrators, and maintainers.

8.3 Fundamental WinWin Concepts

8.3.1 The WinWin Approach

The general win–win approach evolved more or less independently as an interpersonal-relations [17], success-management [10], and project-management [2] approach. We usually define it as "a set of principles, practices, and tools, which enable a set of interdependent stakeholders to work out a mutually satisfactory (win–win) set of shared commitments."

Interdependent stakeholders can be people or organizations. Their shared commitments can relate to information system requirements in particular (the WinWin groupware system's primary focus) or can cover most continuing relationships in work and life (for example, international diplomacy). Mutually satisfactory generally means that people do not get everything they want but can be reasonably assured of getting whatever it was to which they agreed. Shared commitments are not just good intentions but carefully defined conditions. If someone has a conditional commitment, he or she must make it explicit to ensure all stakeholders understand the condition as part of the agreement.

The WinWin approach is descriptive, in that the main purpose of the system is to negotiate a set of mutually satisfactory agreements that are foundations to requirements, constraints, and plans of the project.

The WinWin negotiation approach addresses some of the problems related with rationale capture. It reduces the work required to gather rationale by providing a well-defined structure and process to negotiate. In addition, the negotiation allows all success-critical stakeholders to participate the process where both recorders and users of the rationale are involved. The process also makes it easy to collect and share the rationale behind the decisions made. Stakeholders using the system simultaneously make rationale capture easier and faster. Rationale generated during negotiation is captured within EasyWinWin. The brainstorming statements are attached to the resulting win conditions to preserve the brainstorming rationale. Issues are attached to win conditions. The traceability links and the containment relations between elements are used to display the reasoning and knowledge behind the agreements. Moreover, the impact of changing decisions is traceable to the related elements.

Win–Lose Does Not Work

In requirements negotiation, nobody wants a lose–lose outcome. Win–lose might sound attractive to the party most likely to win, but it usually turns into a lose–lose situation. Table 8.1 shows three classic win-lose patterns among the three primary system stakeholders in which the loser's outcome usually turns the two "winners" into losers [6].

Table 8.1. Frequent software evelopment win–lose patterns

Proposed solution	Winner	Loser
quickly build a cheap, sloppy product	developer and customer	user
add lots of bells and whistles	developer and user	customer
drive too hard a bargain	customer and user	developer

As the table shows, building a quick and sloppy product might be a low-cost, near-term win for the software developer and customer, but the user (and maintainer) will lose in the long run. In addition, adding lots of marginally useful bells and whistles to a software product on a cost-plus contract might be a win for the developer and users, but it is a loss for the customer. Finally, "best and final offer" bidding wars that customers and users impose on competing developers generally lead to lowball winning bids, which place the selected developer in a losing position.

However, nobody really wins in these situations. Quick and sloppy products destroy a developer's reputation and have to be redone – inevitably at a higher cost to the customer. The bells and whistles either disappear or (worse) crowd out more essential product capabilities as the customer's budgets are exhausted. Inadequate lowball bids translate into inadequate products, which again incur increased customer costs and user delivery delays to reach adequacy.

Why WinWin Works

Builds Trust and Manages Expectations

If stakeholders consistently find other stakeholders asking about their needs and acting to understand and support them, they will end up trusting each other more. In addition, if they consistently find them balancing their needs with other stakeholders' needs, they will have more realistic expectations about getting everything they want. As they work together to negotiate their requirements, they give the project shape, and their merged visions become a system that all stakeholders can accept. If, on the other

hand, stakeholders do not negotiate together, there is little chance the resulting system will accommodate their needs, and the project will fail.

Helps Stakeholders Adapt to Changes in the Environment that Affect Requirements
Instead of rigorous requirements in ironbound contracts, doing business in Internet time requires stakeholders with a shared vision and the flexibility to quickly renegotiate a new solution once unforeseen problems or opportunities arise [3]. A WinWin approach builds a shared vision among stakeholders and provides the flexibility to adapt to change.

Helps Build Institutional Memory
The decisions, the why behind the what, that lead to a work result often vanish. By capturing and preserving stakeholder negotiations, WinWin supports long-term availability of the decision rationale and thus helps build institutional memory. Having more auditable decisions creates more detailed, accurate, and complete deliverables.

8.3.2 How Does the WinWin Negotiation Model Work

Key activities of WinWin negotiation model include (1) the identification of success-critical stakeholders; (2) the elicitation of the success-critical stakeholders' primary win conditions; (3) the negotiation of mutually satisfactory win-win situation packages (requirements, architectures, plans, critical components, etc.); and (4) value-based monitoring and control of a win-win equilibrium throughout the development process.

The WinWin negotiation model has four main conceptual artifacts: *Win condition*: capturing the desired objectives and constraints of the stakeholder; *Issue*: capturing the conflict between win conditions and their associated risks and uncertainties; *Option*: capturing a decision choice for resolving an issue; *Agreement*: capturing the agreed upon set of win conditions which satisfy stakeholder win conditions and/or capturing the agreed options for resolving issues.

The negotiation model guides success-critical stakeholders in elaborating mutually satisfactory agreements. Stakeholders express their goals as win conditions. If everyone concurs, the win conditions become agreements. When stakeholders do not concur, they identify their conflicted win conditions and register their conflicts as issues. In this case, stakeholders invent options for mutual gain and explore the option trade-offs. Options are iterated and turned into agreements when all stakeholders concur. It is important to notice that open, unresolved issues represent potential project

risks or conflicts that need to be addressed. Additionally, a *domain taxonomy* is used to organize WinWin artifacts, and a glossary captures the domain's important *terms*. The stakeholders are in a WinWin equilibrium state when the agreements cover all of the win conditions and there are no outstanding issues (see Fig. 8.2). The negotiation proceeds until all of the stakeholders' win conditions are entered and the WinWin equilibrium state is achieved, or until the stakeholders agree that the project should be disbanded because some issues are irresolvable. In such situations, it is much preferable to determine this before rather than after developing the system.

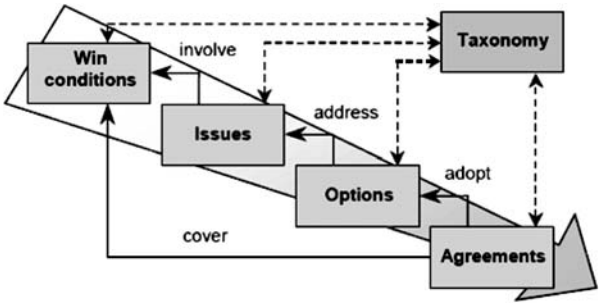

Fig. 8.2. The WinWin negotiation model

The WinWin negotiation model aims at coordinating decision-making activities made by various stakeholders in the software development process. It belongs to the category of supporting collaboration described in Sect. 1.4.1. It guides success-critical stakeholders through a process of eliciting, elaborating, prioritizing, and negotiating requirements. It also provides the support for future changes by keeping the traceability of the artifacts and their rationale.

The negotiation process supports the engineering and management activities of rationale capture. The artifacts and their rationale captured during requirements negotiation shapes the decision made through the software development. In addition, the artifacts provide additional information to check the project status and manage the project risks. The higher number of issues identified and resolved helps reduce risks early in a project and the chances of it derailing later.

The rationale capture during negotiation improves the communication between stakeholders and the quality of the products. Rationale on the negotiation results supports communication between all success-critical stakeholders.

8.4 Tool Support for WinWin Requirements Negotiation

EasyWinWin is a requirements negotiation methodology that combines the WinWin Spiral Model of Software Engineering from USC's Center for Software Engineering with state-of-the-art collaborative knowledge techniques and automation of a Group Support System (GSS) from GroupSystems.com. A GSS is a suite of software tools that can be used to create, sustain, and change patterns of group interaction in repeatable, predictable ways [14].

EasyWinWin helps a team of stakeholders to gain a better and more thorough understanding of the problem and supports cooperative learning about other's viewpoints. Moreover, it helps increase stakeholder involvement and interaction. EasyWinWin defines a set of activities guiding stakeholders through a process of gathering, elaborating, prioritizing, and negotiating requirements. The nominal purpose of the EasyWinWin methodology is to create an acceptable set of system requirements. Teams can use EasyWinWin throughout the development cycle to create a shared project vision, to develop high-levels requirements definition, to produce detailed requirements for features, functions, and properties, COTS acquisition and integration, COTS product enhancement, and to plan requirements for transitioning the system to the customer and user.

The negotiation model provides the capture, representation, and use of rationale. Rationale is captured during stakeholders' communication and negotiation in a structured way in which the relations between the artifacts are clear to the stakeholders. The tool provides the distribution of rationale feature for concurrent user. It is both easy to capture, modify, and review rationale during negotiation. It increases collaboration and coordination with group awareness, synchronous and asynchronous modes of communication, and support for trade-off analysis. Rationale used during and after negotiation to agree on the development artifacts. However, the tool doesn't provide support for rationale preservation and interfaces to legacy components currently because of the reason it is being used for requirements negotiation.

8.4.1 The Negotiation Process

The input to an EasyWinWin workshop is typically a mission statement outlining the high-level objectives of a project and another statement specifying the negotiation purpose, i.e., the objectives of a negotiation within a project. In each activity in this process the team adds details and increases precision. The EasyWinWin process is comprised of the following activities:

Review and expand negotiation topics. Stakeholders jointly refine and customize an outline of negotiation topics based on a taxonomy of software requirements. The shared outline helps to stimulate thinking, to organize negotiation results, and serves as a completeness checklist for negotiations.

Brainstorm stakeholder interests. Stakeholder share their goals, perspectives, views, background, and expectations by gathering statements about their vested interests.

Converge on win conditions. Stakeholders jointly craft a list of clearly stated, unambiguous win conditions by considering and discussing all ideas contributed in the brainstorming session.

Capture a glossary of terms. Stakeholders define and share the meaning of important terms of the project in a glossary.

Prioritize win conditions. Stakeholders prioritize the win conditions to define the scope of work and to gain focus.

Reveal issues and constraints. Stakeholders surface and understand issues.

Identify issues and options. Stakeholders surface the issues that arise due to constraints, risks, uncertainties, and conflicting win conditions. They propose options to resolve these issues.

Negotiate agreements. Stakeholders negotiate mutual commitments by considering win conditions that raised no issues and all proposed options.

8.4.2 The Negotiation Process Deliverables

The activities of the EasyWinWin process are summarized above and shown in Fig. 8.3 with related work products (for a more detailed description please refer to [8, 12]). The results of each activity in the process is a well-defined deliverable (1) negotiation topics organized in a domain taxonomy, (2) a glossary defining key project terms, (3) agreements providing the foundation for further plans, (4) open issues addressing constraints, conflicts, and known problems, as well as (5) further rationale showing the negotiation history (comments, win conditions, issues, options, etc.).

Major results of the negotiation process are a list of agreements and a list of unresolved issues (e.g., caused by stakeholder dissent), which have to be managed as potential projects risks. Agreements of success-critical stakeholders are input to the project contract and to refinement during requirements engineering activities. The WinWin tree shows how agreements and open issues can be traced back to stakeholder win conditions.

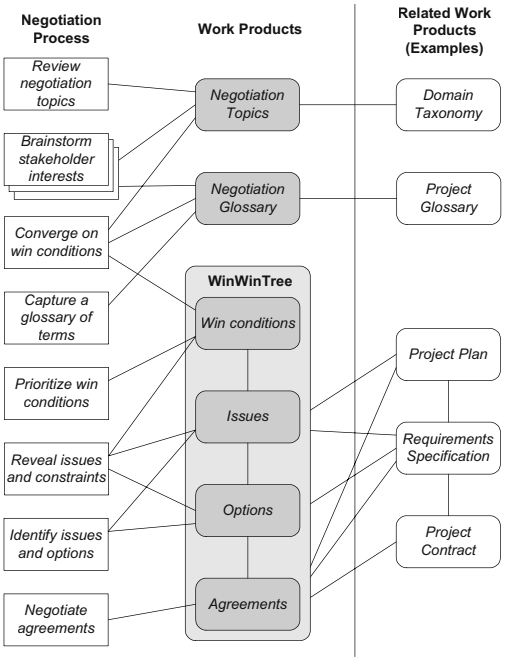

Fig. 8.3. EasyWinWin activities and work products with relationships to important work products in the software life-cycle

8.5 An Example – Using WinWin in Software Development

EasyWinWin has been used in more than 100 real-world projects in various domains (e.g., digital libraries, e-marketplace, and collaboration technology). By using the MBASE approach throughout the software development, we find that using a WinWin requirements negotiation approach helps stakeholders prioritize their requirements and capture the rationale for their decisions.

One of project from the real-world projects that we will use as an example is as follows:

"Information Services Division (ISD) would like to replace its current timecard and timesheet (paper) system with an electronic, web-based system to simplify data collection, to more accurately record hours worked for all its employees, and to provide personnel management tools for supervisors and directors."

The EasyWinWin workshop started with the team reviewing and expanding the negotiation topics based on domain taxonomy which helped

organize the artifacts that emerge later in the process. Figure 8.4 shows part of the taxonomy for our project.

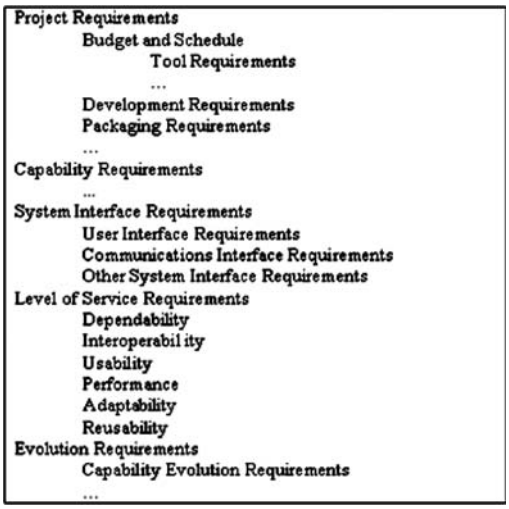

Fig. 8.4. Part of the domain taxonomy

Then stakeholders brainstormed on the project and contributed their interests. Some examples are:

- The system must provide some benefit to the employees that are using it – such as providing them with their current vacation balance.
- System should have capability to correct errors on previous timecards.
- Hierarchical structures can provide several levels of access for different management groups.
- Some ISD constituents would prefer a card swipe or biometric clock-in/-out system connected to the network. Supervisors feel this will reduce clocking-in/-out for others.

The resulting collection of stakeholder statements and ideas provided a starting point and rationale for elaborating win conditions and defining important terms of the project domain. The brainstorming statements were attached to resulting win conditions to preserve brainstorming rationale.

After that, stakeholders voted on each win condition according to two criteria: *Business Importance* and *Ease of Implementation*. During this activity, developers typically focused on technical issues, while clients and users concentrated on the business relevance (see Fig. 8.5).

In the next step, stakeholders examined the results of the prioritization and identified issues and options in several iterations. During the revealing

issues and constraints, the stakeholders modified the priorities according to the updated information they got. The WinWin equilibrium state holds when all win conditions are covered by agreements, and there are no outstanding issues. As soon as some stakeholder enters an issue and an associated conflicting win condition, the negotiation leaves the WinWin equilibrium state, and the stakeholders attempt to formulate options to resolve the issue. For example, if the conflicting win conditions are to have the system run on a Windows platform and a UNIX platform, an acceptable option might be to build the system to run on a Java Virtual Machine.

	Win Conditions	Business Importance	Ease of Implementation
1.	Application Capabilities		
1.1	W7 System should have the capability to generate reports required by th	8.80	6.60
1.2	W8 System should have multiple levels of security	8.60	5.40
1.3	W10 Rules governing vacation/sick time should be programmable into t	8.60	5.20
1.4	W19 System shall be capable of validating input	7.00	7.80
1.5	W20 System shall be firewalled from outside of USC	8.80	5.40
1.6	W30 Proof of concecpt should be able to support maximum of 30 users	8.00	6.40
1.7	W24System shall handle multiple users concurrently	8.20	6.20
2.	Interfaces		
2.1	W12 System should support web-based and card swipe interfaces.	6.20	5.60
2.2	W17 User interface shall be able to support Internet Explorer and Netsc	8.40	8.60
2.3	W18 The output of the system shall be compatible with the Payroll accou	8.20	6.40
3.	Level of Service		
3.1	W2 System shall reduce time and effort required to process payroll hour	8.60	5.80

Fig. 8.5. Some voting results of Win Conditions – *gray* as consensus, *black* as lack of consensus

The WinWin Tree has all the information gathered during the requirements negotiation: Unique numbers for artifacts that help tracing the artifacts through the software's life cycle, priorities for win conditions, stakeholders who identified issues and options, and the taxonomy elements those artifacts belong to. The WinWin Tree also captures the rationale for win conditions and how stakeholders reach an agreement by including all proposed options, whether adopted or not, all issues which eventually addressed and all win conditions (see Fig. 8.6).

The negotiation results, mainly agreements, become the foundations of requirements whereas the other artifacts are the rationale for further decisions made during the development life cycle such as major risks, iteration plans, etc. Agreements that cover the lower-priority win conditions become evolution requirements, providing the basis for architecting the system to easily drop them (if necessary to meet the schedule) or incorporate them in later increments. For example, "W12 [FGT] System should support web-based and/or card swipe interfaces. [Taxonomy 3.1]" belonged to User Interface Requirements during the negotiation. However, after the prioritization it is categorized as not important and very difficult to implement. An issue identified by the administrator as "I12 Supporting card swipe interfaces requires additional hardware purchase and integra-

tion [Administrator] [Taxonomy 3.1]". This issue occurred because of a budget limit for the project. So the stakeholders provided an option to have web-based user interface first, and left the card swipe interface as a technology evolution requirements. At the end there were two requirements (1) Web-based interface as interface requirements, (2) adding new input devices as magnetic card readers as an evolution requirement.

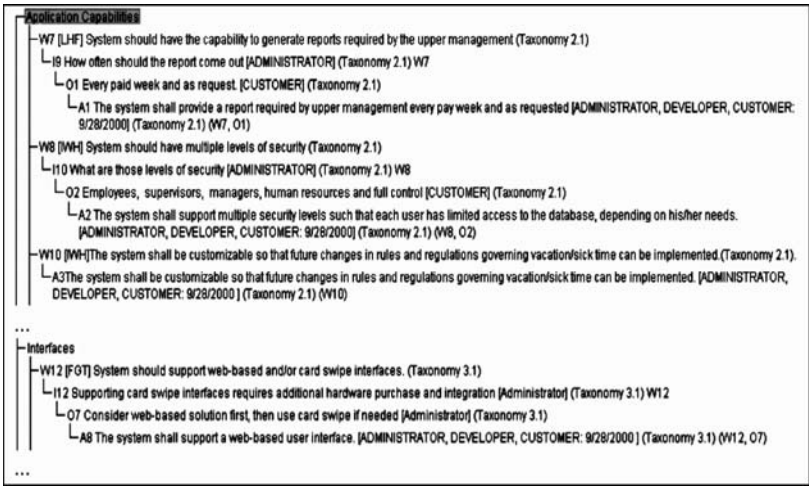

Fig. 8.6. A small part of WinWin Tree – initial capability and interface sections

The team using EasyWinWin is able to develop a broader and deeper set of results in a shorter time. An EasyWinWin negotiation results in a significantly higher number of artifacts compared to traditional paper or blackboard-based approaches: our experience to date shows that typical negotiations about system requirements with 10+ stakeholders result in 300+ brainstorming ideas, 100+ win conditions, 50+ issues, 50+ options, and 100+ agreements in less time than other traditional techniques. Even though, the teams had a similar educational background and basically the same win conditions, they came up with very different negotiation approaches and solutions.

The focus on consensus leads to a higher acceptance of decisions and to an increased mutual understanding among the involved parties. The evaluation of the WinWin model shows that the use of an issue model for negotiation support enhances trust and shared understanding among shareholders, even in the presence of uncertainties and changing requirements.

8.6 Using the Captured Rationale to Improve Later Decisions

Design rationale is documented in the WinWin artifacts to provide a corporate memory. Risks are explicitly addressed in WinWin to pinpoint possible breakdowns and propose early fixes. This makes it easier to increment and evolve requirements in the spiral model.

As the project unfolds, the WinWin results are useful in many ways. First and foremost it is the highest-level expression of requirements. All subsequent requirements specifications refer back to the WinWin results. This provides an answer to the often-asked questions, "Where did these requirements come from? Why were they adopted? Which requirements satisfy which needs of which stakeholders? Who will be affected if we change the specification?" The WinWin results provide a common reference point for organizing management procedures, cost estimates, schedules, etc.

An initial developer win condition was to save development time and money by reusing a research planning module. An issue entered by the maintainer indicated that module would be risky and expensive to maintain. An agreement to drop the win condition was recorded. Later, the project got behind schedule and the new developer manager proposed to recover by reusing the research planning module. Without the captured rationale, the project would have done this and caused major maintenance problems. With the captured rationale, the developers can check the status of the planning module and reject its use if it is still risky and expensive to maintain.

Thus, capturing the rationale behind the decisions generated by the WinWin negotiation enables the stakeholders to avoid mistaken decisions often associated with personnel turnover. This allows the designers to accommodate a much broader set of needs, and allows the stakeholders to negotiate trade-offs with one another based on well understood interest. WinWin also makes it far easier to modify requirements part-way through the project as new constraints are discovered because every requirement can be tied back to some set of win conditions, which in turn tied back to some set of stakeholders. 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.

8.7 Related Work

Rationale capture models for software requirements and design decisions capture dependencies between multiple stakeholder objectives, issues, requirements, design, and trade-off options. IBIS addresses multistakeholder consideration by supporting relations among system objectives. Issues can be viewed as requirements that impact on design decisions. Conklin et al. [9] attempted to allow less disruption to the design process with a graphical tool, gIBIS, to record the rationale. Although IBIS structures support analysis of requirements interactions, no tools are provided for analyzing trade-offs, so the design decision may overlook optimal solutions. There is also no negotiation strategy embedded to reconcile different perspectives. The WinWin approach is specifically for recording architectural rationale. While both gIBIS and WinWin attempt to reduce the overhead in capturing rationale, they focus on particular elements that must still be formally documented during the discussions.

The WinWin approach is aimed to provide not as much structure as attempted in gIBIS, SIBYL, and REMAP, which have difficulties in scaling up to large systems. However, the Win–Win Spiral Process model and WinWin are also trying to provide stronger support for scalable shared ontologies and for collaboration objectives via the domain taxonomy and via the conceptual bases for collaboration and software development provided by Theory W and the Spiral Model. For example, the objective of achieving a win–win situation among stakeholders' win conditions provides a much more explicit answer to the question, "What are we trying to collaborate about?" than other conceptual frameworks for collaboration.

8.8 Future Directions

EasyWinWin helps smooth the transition from WinWin stakeholder agreements to requirements specifications. Mapping the WinWin domain taxonomy onto the table of contents of the requirements specification and requiring the use of the domain taxonomy as a checklist for developing WinWin agreements effectively focused stakeholder negotiations. But the result of a WinWin negotiation is typically not a complete, consistent, traceable, testable requirements specification. For example, stakeholders may become enthusiastic about proposed new capabilities and ratify idealistic agreements such as "anytime, anywhere" service. We are exploring how to automate parts of the requirements transition to make it even smoother. For rationale capture, further formatting and indexing capabilities need to be researched and experimented with to capture rejected as well as accepted win conditions and options. Also, some research

capabilities are experimented with rationale capture such as audio or video clips are now becoming economically feasible to incorporate.

References

[1] Boehm B (1988) A spiral model of software development and enhancement. Computer, 21(5): 61–72
[2] Boehm B (1996) Anchoring the Software Process. IEEE Software, 13(4): 73-82
[3] Boehm B (2000) Requirements That Handle IKIWISI, COTS, and Rapid Change. Computer, 33(7): 99-102
[4] Boehm B, Bose P (1994) A Collaborative Spiral Software Process Model Based on Theory W. Proc. Int'l Conf. Software Process, IEEE CS Press, Los Alamitos, California, pp 59-68
[5] Boehm B, Ross R (1989) Theory W software project management: Principles and examples. IEEE Trans. Softw. Eng., 15(7): 902–916
[6] Boehm B, Bose P, Horowitz E, Lee MJ (1994) Software Requirements as Negotiated Win Conditions. In: Proceedings of the First International Conference on Requirements Engineering. IEEE Computer Society, Colorado Springs CO, pp. 74–83
[7] Boehm B, Abts C, Brown AW, Chulani S, Clark BK, Horowitz E, Madachy R, Reifer D, Steece B (2000) Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River, NJ
[8] Boehm B, Gruenbacher P, Briggs RO (2001) Developing Groupware for Requirements Negotiation: Lessons Learned. IEEE Softw., 18(3): 46–55
[9] Conklin J, Begeman M (1988) gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Trans. Off. Inform. Syst., 6(3): 303–331
[10] Covey S (1990) The Seven Habits of Highly Effective People. Fireside Books, New York
[11] Fisher R, Ury W (1981) Getting To Yes. Houghton-Mifflin, Boston
[12] Gruenbacher P (2000) EasyWinWin OnLine: Moderator's Guidebook. GroupSystems.com and USC-CSE
[13] Lee J (1990) SIBYL: A Qualitative Decision Management System. In: Winston P and Shellard S (eds.) Artificial Intelligence at MIT; Expanding Frontiers, MIT Press, Cambridge, pp. 106–133
[14] Nunamaker J, Briggs R, Mittleman D, Vogel D, Balthazard P (1996) Lessons from a dozen years of group support systems research: A discussion of lab and field findings. J. Manage. Inform. Syst., 13(3):163–207
[15] Ramesh B, Dhar V (1992) Supporting Systems Development by Capturing Deliberations during Requirements Engineering. IEEE Trans. Softw. Eng. 18(6): 498–510
[16] The Standish Group (1995) CHAOS Report
[17] Waitley D (1985) The Double Win. Berkeley Books, New York