

# Lessons Learned from Four Generations of Groupware for Requirements Negotiation

Barry Boehm

*Computer Science Department  
University of Southern California  
941 W. 37th Place  
Los Angeles, CA 90089-0781  
boehm@sunset.usc.edu*

Paul Grünbacher

*Systems Engineering & Automation  
Johannes Kepler University Linz  
Altenbergerstr. 69  
4040 Linz, Austria  
pg@sea.uni-linz.ac.at*

Robert O. Briggs

*GROUPSYSTEMS.COM  
1430 E. Fort Lowell Rd. #301  
Tucson, AZ 85719,  
bbriggs@groupsystems.com*

## 1 Introduction

Requirements definition is a complex and difficult process and defects in defining the requirements often lead to costly project failures [StandishGroup-1995]. There is no complete and well-defined set of requirements waiting to be discovered in system development. Different stakeholders – users, customers, managers, domain experts, and developers – come to the project with diverse expectations and interests. Requirements emerge in a highly collaborative, interactive, and interdisciplinary negotiation process that involves heterogeneous stakeholders.

USC's Center for Software Engineering, initially with its Affiliates' support, later with DARPA and Air Force Research Labs' support and a grant by the Austrian Science Fund (second author), developed a series of four groupware implementations for the WinWin requirements negotiation approach. These reflect increasing understanding of what was needed for successful WinWin groupware operations and technology support. Groupware-supported methodologies are among the hardest to get right. The rapidly moving technology of distributed interactive systems is a major challenge. However, even bigger is the challenge of creating a system that works well with people of different backgrounds, in different places, and often at different times. Collaborative technology supporting requirements negotiation has to address the heterogeneity of stakeholders in particular.

Here we present the major lessons we have learned in developing the four generations of a distributed groupware system called WinWin. We also summarize the degree to which the current system, EasyWinWin, satisfies the original and evolving objectives for such a system, and summarize how it enables and facilitates stakeholder participation and collaboration.

## 2 WinWin requirements negotiation approach

The original motivation for a WinWin groupware system was the first author's frustration in using a manual WinWin approach to manage large projects at DARPA. For example, WinWin management of the \$100 million DARPA STARS program was done primarily via monthly meetings of many STARS stakeholders: 3 prime contractors and their 3 commercial counterparts; 3 user representatives from the Army, Navy, and Air Force; DARPA customers, contract managers, and several research and support contractors. Each meeting would end up with a WinWin agreement that felt like three steps forward. However, by the next meeting, we would take two steps back, as the distributed stakeholders independently "re-interpreted" the agreements. As a result, it took six months to achieve a shared vision documented by the prime contractors' success plans. Our analysis at the time indicated that a WinWin groupware support system could have reduced this to 1-2 months.

The WinWin approach evolved more or less independently as an interpersonal relations [Waitley-1985], success management [Covey-1990], and project management [Boehm-1989] approach. A reasonable common definition is:

*The WinWin approach is 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.*

The interdependent stakeholders can be either people or organizations. Their shared commitments can be about information system requirements (the primary focus of the WinWin groupware system), but can cover any continuing relationships in work and life. "Mutually satisfactory" generally means that people do not get everything they want, but that they can be reasonably assured of getting what was agreed to. "Shared commitments" are not just good intentions, but carefully defined conditions. If someone has a conditional commitment, the condition needs to be made explicit, and understood as part of the agreement by all stakeholders.

## 2.1 Why does WinWin work?

### Win-Lose doesn't work

In a requirements negotiation, nobody wants a lose-lose outcome. Win-lose may sound attractive to the party most likely to win, but it usually turns into lose-lose. Table 1 shows three classic win-lose patterns among the three primary system stakeholders – developers, customers, and users – in which the loser's outcome usually makes the two "winners" into losers as well [Boehm-1994].

**Table 1: Frequent Software Development Win-Lose Patterns (which usually turn into lose-lose situations)**

Proposed Solution	"Winner"	Loser
Quick, Cheap, Sloppy Product	Developer & Customer	User
Lots of "bells and whistles"	Developer & User	Customer
Driving too hard a bargain	Customer & User	Developer

Building a quick and sloppy product may be a low-cost, near-term win for the software developer and customer, but it will be a lose for the user (and the maintainer). Adding lots of marginally useful "bells and whistles" to a software product on a cost-plus contract may be a win for the developer and users, but is a lose for the customer. And "best and final offer" bidding wars imposed on competing developers by customers and users generally lead to low-ball winning bids which place the selected developer in a losing position.

Actually, nobody wins in the above 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 low-ball bids translate into inadequate products, which again incur increased customer costs and user delivery delays to reach adequacy.

### WinWin builds trust and manages expectations

If you consistently find other stakeholders asking about your needs and acting to understand and support them, you will end up trusting them more. In addition, if you consistently find them balancing your needs with other stakeholders' needs, you will have more realistic expectations about getting everything you might want.

### WinWin helps stakeholders adapt to changes in the environment

Our traditional, adversarial, lawyer-oriented contracting mechanisms are no match for our current world of increasing rapid change in technology, mergers, reorganizations, and personnel turnover. 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 [Boehm-2000b] [Gause-1989] [Robertson-1999] [Jackson-1995] [Highsmith-2000]. A WinWin approach builds a shared vision among stakeholders, and provides the flexibility to adapt to change.

### WinWin helps to build institutional memory

The *why* behind the *what*, i.e., the decisions that led to a work result often vanish. By capturing and preserving stakeholder negotiations WinWin supports long-term availability of the decision rationale and thus helps to build institutional memory. Decisions are more auditable and result in more detailed, accurate, and complete deliverables.

## 2.2 How does the WinWin approach work?

The particular WinWin approach we have evolved includes a negotiation model for converging to a WinWin agreement, and a WinWin equilibrium condition to test whether the negotiation process has converged.

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. Additionally, a *domain taxonomy* is used to organize WinWin artifacts. Important *terms* of the domain are captured in a glossary.

The stakeholders are in a WinWin equilibrium condition when all of their win conditions are covered by agreements, and there are no outstanding issues.

### **2.3 Four generations of tool support**

The WinWin negotiation model provided the basis of all four implementations of WinWin groupware systems.

#### **First Generation (1G): Initial Prototype**

The first WinWin groupware implementation was a prototype developed in concert with Perceptronics' CACE-PM® support system for concurrent engineering of multi-chip modules. CASE-PM® enabled us to develop a useful rapid prototype, which was useful enough for demonstrations and for an initial experiment. This involved the system's developers role-playing as future system developers, customers, and users negotiating the requirements for a more robust version of WinWin. Performing the WinWin negotiation with Generation 1 of the tool gave us a strong, shared vision for the next version of the system, validating its utility as a groupware capability.

#### **Second Generation (2G): Strong Vision, Not-So-Strong Architecture**

The second-generation WinWin system used a Sun-UNIX client-server architecture, X/Motif GUI support, and its own database server. It was used experimentally by some friendly industry users. Its main value was to identify a number of inconsistencies in the negotiation model and the artifacts, among the artifacts, and between GUI and the database server. We had underestimated how much detailed software engineering was needed to get from a shared groupware vision to a groupware support system.

#### **Third Generation (3G): Muscle-Bound Architecture**

The third-generation WinWin system had a formally analyzed negotiation model, uniform artifact look and feel, carefully defined GUI-database interfaces, and rigorous enforcement of the negotiation model. The 3G WinWin also had a number of amenities for voting, for attaching associated documents or analysis-tool runs, and for big-picture negotiation visualization and navigation. Major problems of 3G were the insufficient robustness of the system and the overly strict enforcement of the negotiation approach resulting in the inflexibility to adapt to different negotiation situations.

#### **Fourth Generation (4G): Group Support System Infrastructure**

These experiences turned USC more toward developing a version of WinWin based on a commercial groupware infrastructure developed by GroupSystems.com in cooperation with the University of Arizona [Nunamaker-1997]. Our current collaboration between USC and GroupSystems.com has led to a fourth-generation system, called EasyWinWin [Gruenbacher-2000, Boehm-2000].

### **3 EasyWinWin**

EasyWinWin is a requirements definition approach that is based on a Group Support System to enable the involvement and interaction of key stakeholders. A Group Support System (GSS) is a suite of software tools that can be used to create, sustain, and change patterns of group interaction in repeatable, predictable ways (see GSS sidebar for details).

EasyWinWin defines a set of activities guiding stakeholders through a process of gathering, elaborating, prioritizing, and negotiating requirements. EasyWinWin uses group facilitation techniques that are supported by collaborative tools. Table 2 summarizes the EasyWinWin steps and group techniques.

**Table 2: EasyWinWin activities, group techniques, and patterns of thinking**

<i>Activity</i>	<i>Purpose</i>	<i>Group Technique</i>	<i>Thinking Pattern (see Sidebar)</i>
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 win conditions, and serves as a completeness checklist for negotiations.	Stakeholders add comments recommending changes to the outline.	Diverge
		A moderator reviews these comments together with the group and modifies the outline itself.	Converge
Brainstorm stakeholder interests	Stakeholder share their goals, perspectives, views, background and expectations by gathering statements about their win conditions.	Free-format brainstorming: Anonymous, rapid brainstorming on electronic discussion sheets.	Diverge
Converge on Win Conditions	Stakeholders jointly craft a non-redundant list of clearly stated, unambiguous win conditions by considering all ideas contributed in the brainstorming session	FastFocus: A structured discussion to converge on key win conditions.	Converge
		Categorization of win conditions into negotiation topics (Figure 1)	Organize
Capture a glossary of Terms	Stakeholders define and share the meaning of important terms of the project/domain in a glossary of terms.	Stakeholders propose initial definition of terms based on stakeholder statements. The team then jointly reviews and agrees on the terms.	Elaborate
Prioritize Win Conditions	The team prioritizes the win conditions to define the scope of work and to gain focus.	Stakeholders rate win conditions for each of two criteria: - Business importance: relevance of a win condition to project/company success - Ease of realization: perceived technical or economic constraints of implementing a win condition	Evaluate
Reveal Issues and Constraints	Stakeholders surface and understand issues.	Crowbar: Analyze prioritization poll to reveal conflicts, constraints, different perceptions, etc. (Figure 2)	Build consensus
Identify Issues, Options, Agreements	Identify the issues that arise due to constraints, conflicting win conditions. Propose Options to resolve these issues. Negotiate agreements.	WinWinTree: Develop/Review pass for issues, options	Elaborate
		Negotiation of agreements	Build consensus

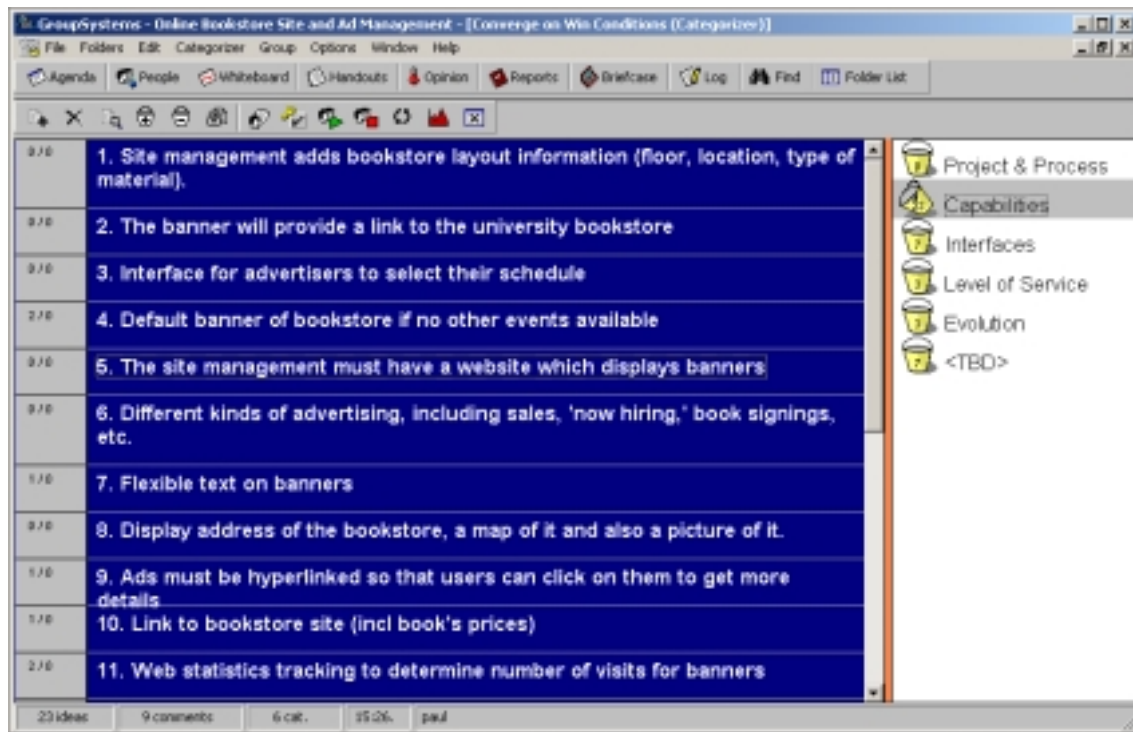


Figure 1: Team builds a clean list of win conditions and organizes win conditions into pre-defined buckets

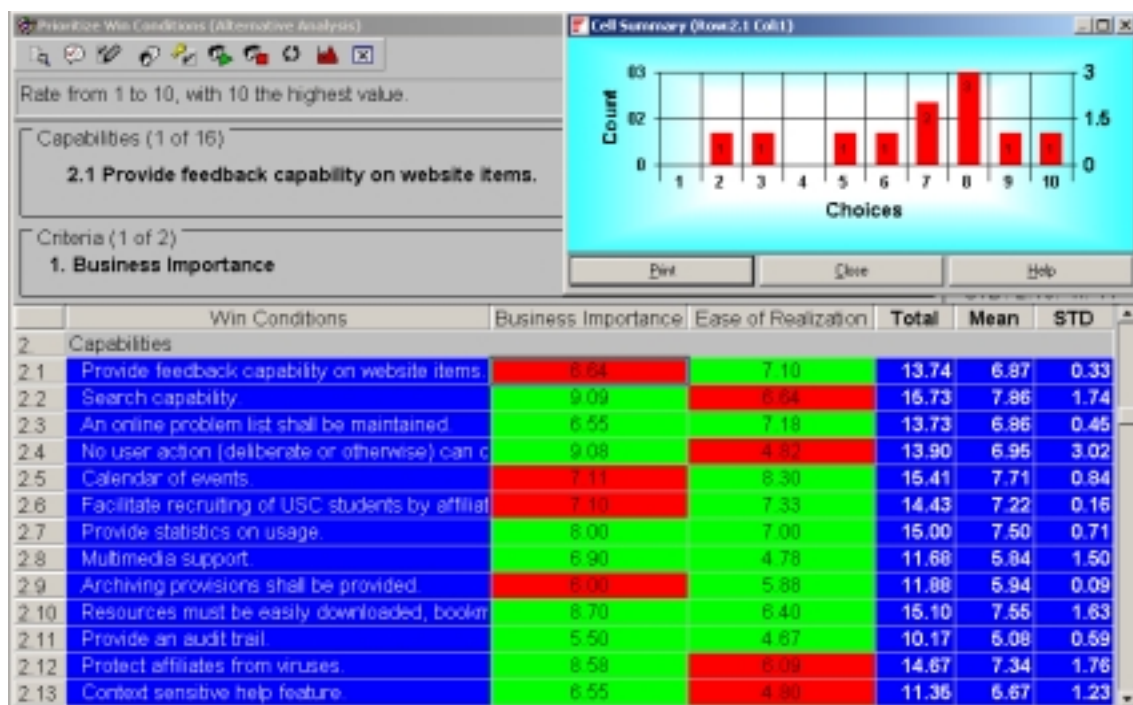


Figure 2: Red cells indicate lack of consensus. Oral discussion of voting pattern displayed as cell graph reveals unshared information, unnoticed assumptions, hidden issues, constraints, etc. Sample cell graph shows voting pattern for Win Condition 2.1, criterion Business Importance

EasyWinWin has been used in about 30 real-world projects to date. We applied the approach in various domains (e.g., digital libraries, e-Marketplace, collaboration technology) and thoroughly explored and refined the various collaborative techniques with the goal of streamlining the negotiation protocols and the overall order and design of process steps. Our experiences have been captured in a detailed process guidebook, explaining the approach to a project manager or facilitator [USC-1999].

## **4 Lessons Learned**

We will discuss three major types of lessons learned in the development of the four generations of WinWin systems: (1) methodology lessons (2) groupware lessons (3) project lessons.

### **4.1 Methodology lessons**

#### **Define a repeatable requirements negotiation process**

The first three generations didn't provide a strategy on how to carry out a concrete WinWin negotiation. In the EasyWinWin project we focused on moving people through a process that builds mutual understanding and shared vision in progressive steps. Each step involves the stakeholders in assimilating each others' views, and in building consensus on a mutually satisfactory (win-win) shared vision and set of system requirements. A process guide explains the use of group techniques and collaborative tools in a requirements negotiation [USC-1999]. We found that a detailed process guide reduces variance in the quality of deliverables and helps lower skilled or less experienced practitioners to accomplish more than would be possible with straight stand-up facilitation.

#### **Incorporate facilitation and collaboration techniques**

A major lesson learned is that collaborative technology for requirements engineering has to be based on collaboration and facilitation techniques that emphasize group dynamics. The first three generations of WinWin environments emphasized modeling constraints over group dynamics and collaboration support. Groupware-supported collaboration techniques adopted in EasyWinWin allow to create desired patterns of group interaction (see Sidebar).

An example is the anonymous submission of stakeholder contributions like win conditions or voting ballots used to foster candor. People with power differentials can lay it on the line without threat to job, status, relationships, and political position. Increased openness also helps to get to the root issues in a hurry. People up and down the hierarchy can find out what is really going on thus avoiding the Abilene Paradox (people agree to an unattractive option because they erroneously believe it will make the option-proposer happy) [Harvey-1974].

#### **Recognize the role of negotiations in the life-cycle**

A major lessons learned from the experiment with the 1G WinWin system was [Boehm-1994] that the WinWin approach helps bridge a previous gap in using the Spiral process model: how to determine the next round of objectives, alternatives, and constraints. This led to the WinWin Spiral Model extensions [Boehm-1994, Boehm-1998], now used by several organizations.

Experiments also showed that prototyping should be performed ahead of and during requirements negotiations: The 3G WinWin was sufficiently robust to have supported four years' work of 15-20 project negotiations per year [Boehm-1998, Egyed-1999]. These involved USC librarians and student teams negotiating the requirements for operational USC digital library systems, which the student teams then built and transitioned to library use. In the first year, we learned not to do the WinWin negotiations ahead of the prototype, as we rediscovered the IKIWISI (I'll know it when I see it) syndrome. Once the librarians saw the prototype, they wanted to re-do all the negotiations. In the following years, we verified across over 100 requirements negotiations that 3G WinWin was able to support rapid definition and development of unprecedented applications.

WinWin has been successfully used in various contexts of a requirements definition: This includes the development of a shared vision among stakeholders, requirements definition for custom development projects, COTS acquisition and integration, transition planning, as well as COTS product enhancement and release planning.

## **Make sure your stakeholder negotiators are representative, empowered, knowledgeable, collaborative, and committed**

These characteristics came out of our critical success factor analysis of which digital library projects have succeeded or failed in being actually used. It often involves pre-screening of stakeholder negotiations, and shared-knowledge-building activities such as preliminary teambuilding sessions and concurrent prototyping.

## **Refine agreements into more measurable requirements**

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. Rather than dampen their enthusiasm with a precise-wordsmithing exercise, it is better to have a facilitator post-process such agreements into more precise and realistic requirements for the stakeholders to review and iterate.

## **4.2 Groupware lessons**

### **Make use of unobtrusive and flexible tools**

Each of our first three generations of WinWin groupware was increasingly strict about enforcing modeling conventions at the expense of group dynamics.

3G WinWin was also used experimentally by several industry and government organizations. However, this use did not lead to the system’s crossing the chasm into mainstream use. The main reason cited by the mainstream users was that 3G WinWin’s integrity rules were too rigorous. For example, when an Agreement was put to a vote, all of its associated Win Conditions, Issues, and Options were locked to preserve integrity of the voting process. Then, to fix a typo in an artifact that was locked for voting, users had to make all the locked artifacts inactive and copy their contents into a new set of artifacts. Further, one could not define Issues without Win Conditions as referents, or Options without Issues as referents. Thus, the software got in the way of the human interactions that are a critical part of negotiation. In 4G we decided to relax such constraints.

### **Define negotiation model**

Experiments with the 1G WinWin system showed that software requirements negotiation required considerably more database and relationship management than was needed for multi-chip modules. This led to a much more thorough definition of WinWin artifacts and relationships, including the basic negotiation model discussed above.

### **Focus on ease-of-use to foster stakeholder involvement**

People participating in a requirements negotiation typically don’t have the time to take a training before starting to negotiate. Ease of use allows direct participation of more people and elicits more from everybody involved. This leads to better buy-in as more interests can be accommodated earlier in the process. It also helps to develop broader and deeper deliverables: We found that EasyWinWin results surpass 3G WinWin results in terms of number of artifacts that are collected in a negotiation [Egyed-1999]. The higher number of issues identified and resolved helps to reduce risks early in a project and to avoid derailing it later. For example, our 3G WinWin digital library requirements negotiations involved 15-25 win conditions and converged in 2-3 weeks (as compared to 2-3 months for comparable manual win-win requirements negotiations). Similar 4G EasyWinWin requirements negotiations involved 50-120 win conditions and converged in 2-3 days. Further, we found that the stakeholders’ experience with 4G WinWin led to better mutual understanding and greater stability of the negotiated requirements.

### **Provide robust infrastructure**

The 3G homebrew database server was very fragile and prone to lose people’s work in crashes. Using a reliable infrastructure is critical to avoid frustration and to ensure stakeholder buy-in.

### **Provide interoperability**

3G WinWin did not interoperate well with other groupware systems, even after we built an applications program interface for it. One industry project successfully built and applied its own WinWin overlay on top of the groupware system it was using, but did not try to develop a more general capability.

## **Support for multiple modes of interaction**

Beyond same-time/same-place stakeholder interaction, we successfully included remote participants into EasyWinWin workshops by utilizing the web-based capabilities of GroupSystems and additional use of audio links. We are currently elaborating recommendations for geographically distributed teams intending to adopt EasyWinWin activities in different time/different place settings.

### ***4.3 WinWin project lessons***

#### **Involve mainstream end-users**

With persistence and focus on your mainstream end users, you can develop groupware systems which both speed up the initial definition process and help stakeholders achieve a shared vision with lasting value across their application's entire life cycle.

A similar conclusion holds with respect to the representativeness of the user base for which you are building the groupware system. Once we had an annual set of USC projects to support with 3G WinWin, we over-focused on USC users rather than our primary target of mainstream industry users.

#### **When developing groupware, perseverance pays off**

Do not overreact to initial negative experiences. Groupware systems must be carefully balanced to accommodate the many needs of the many stakeholders. In the design of 2G and 3G WinWin, we reacted to the 1G WinWin experience with its high-priced commercial infrastructure by building homebrew infrastructure. The 4G EasyWinWin overlay above GroupSystems' infrastructure has been much more successful.

In the design of 3G WinWin, we also overreacted to some instances of artifact misuse in 2G WinWin by creating a system whose rules were so rigorous that they turned off most users. 3G WinWin improved over 2G WinWin with its well-defined architectural interfaces, but lost out because of its inflexibility for mainstream stakeholder groups.

Relative to the "build it twice" guidance in Royce's initial waterfall-model article [Royce-1970] and in Brooks' Mythical Man Month book [Brooks-1975], one must also add Brooks' "second system syndrome:" that developers, particularly for groupware, are likely to react over-ambitiously to experiences with initial prototypes or systems.

#### **Use the system to plan its own future**

It provides both a good test of the current groupware system and a good way of achieving a shared vision of its future directions. Both USC's experience with 1G WinWin and GroupSystems.com's experience with EasyWinWin substantiates this.

## **5 Conclusions**

Developers of group support systems should not expect to get them right the first time – or even the second time. Our experience with the four generations of WinWin requirements negotiation systems is that it takes several iterations of operational group support systems to start to fully realize their benefits. Even now, we are involved in a fifth iteration to provide better support for less-experienced facilitators. However, the payoffs are worth it: we have experienced about a factor of 4 improvement in multi-stakeholder requirements negotiation time when going from manual negotiation to the 2G-3G WinWin system; and another factor of 5 in going from 2G-3G WinWin to the 4G EasyWinWin system. In addition, the negotiation results have become more thorough and better internalized by the stakeholders.

## **6 References**

- [Boehm-1989] Boehm B., Ross R., Theory W Software Project Management: Principles and Examples, IEEE Transactions on Software Engineering, July 1989, pp. 902-916.
- [Boehm-1994] Boehm B., Bose P., Horowitz E., Lee M.J., Software Requirements as Negotiated Win Conditions, Proceedings Intl. Conf. Rqts. Engineering, IEEE April 1994.
- [Boehm-1998] Boehm B., Egyed A., Kwan J., Port D., Shah A., Madachy R., Using the WinWin Spiral Model: A Case Study, IEEE Computer, 7:33-44, 1998.
- [Boehm-2000] Boehm B., Grünbacher P., Supporting Collaborative Requirements Negotiation: The EasyWinWin Approach, International Conference on Virtual Worlds and Simulation, San Diego, SCS 2000.
- [Boehm-2000b] Boehm B., Requirements that Handle IKIWISI, COTS, and Rapid Change, IEEE Computer, July 2000, pp. 99-102.

- [Brooks-1975] Brooks F.P., *The Mythical Man-Month*, Reading, Mass: Addison-Wesley, 1975.
- [Briggs, DeVreede, Nunamaker, & Tobey, 2001] Thinklets: Achieving Predictable Repeatable Patterns of Group Interaction with Group Support Systems (GSS) by Robert O. Briggs, Gert-Jan de Vreede, and Jay F. Nunamaker, Jr., In: *Proceedings HICSS 2001 (Hawaii International Conference On System Sciences)*, IEEE Computer Society.
- [Covey-1990] Covey S., *The Seven Habits of Highly Effective People*, Fireside Books, 1990.
- [Egyed-1999] Egyed A.F., Boehm B., Comparing Software System Requirements negotiation patterns, *Journal for Systems Engineering*, John Wiley & Sons, 1999.
- [Fjermestad, 2000 – 2001] Fjermestad J., R. Hiltz, *Case and Field Studies of Group Support Systems: An Empirical Assessment*, *Journal of Management Information Systems*, in press.
- [Gause-1989] Gause D. Weinberg G., *Exploring Requirements: Quality Before Design*, Dorset House, 1989.
- [Gruenbacher-2000] Gruenbacher P., *Collaborative Requirements Negotiation with Easy WinWin*, 2<sup>nd</sup> International Workshop on the Requirements Engineering Process, Greenwich London, IEEE Computer Society, 2000.
- [Harvey-1974] Harvey J. B., *The Abilene Paradox and Other Meditations on Management* (San Francisco: Jossey-Bass, 1988). The original publication of the Abilene Paradox appeared as: *The Abilene Paradox: The Management of Agreement*, in *Organizational Dynamics*, 1974.
- [Highsmith-2000] Highsmith J., *Adaptive Software Development*, Dorset House, 2000.
- [Jackson-1995] Jackson, M., *Software Requirements and Specifications*, Addison Wesley, 1995.
- [Nunamaker-1997] Nunamaker, J. Briggs, R., Mittleman, D., Vogel, D., Balthazard, P., *Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings*, *Journal of Management Information Systems*, Winter 1996-97, 13(3), 163-207.
- [Robertson-1999] Robertson, S. and Robertson, J., *Mastering the Requirements Process*, Addison Wesley, 1999
- [Royce-1970] Royce W.W., *Managing the Development of Large Software Systems*, *Proceedings of IEEE WESCON*, pp. 1-9, 1970.
- [Shepherd-1996] Shepherd M.M., Briggs R.O., Reinig B.A., Yen J., Nunamaker J.F., Jr. *Invoking Social Comparison to Improve Electronic Brainstorming: Beyond Anonymity*, *Journal of Management Information Systems*. 12(3):155-170, 1995-96.
- [StandishGroup-1995] *CHAOS Report*, 1995
- [USC-1999] *The EasyWinWin Process Guide: USC-CSE and GROUPSYSTEMS.COM*.
- [Waitley-1985] Waitley D., *The Double Win*, Berkeley Books, 1985.

## 7 Acknowledgments

This research is sponsored by DARPA through Rome Laboratory under contract number F30602-94-C-0195, by the Austrian Science Fund (2<sup>nd</sup> author, Erwin Schrödinger Grant 1999/J 1764), and by the affiliates of the USC Center for Software Engineering: Aerospace, Automobile Club of Southern California, C-Bridge, Chung – Ang U. (Korea), Draper Labs, Draper Labs, Electronic Data Systems Corporation (EDS), Federal Aviation Administration (FAA), Fidelity, GDE Systems, Group Systems.Com, Hughes, Institute for Defense Analysis (IDA), Litton Industries, Inc., Lockheed Martin Corporation, Lucent Technologies, Microsoft, Motorola, Inc., Northrop Grumman Corporation, Rational Software Corporation, Raytheon/East, Raytheon/West, SAIC (Science Applications International Corporation), Software Engineering Institute (SEI Carnegie-Mellon University), Software Productivity Consortium (SPC), Sun Microsystems, Telcordia Technologies, The Boeing Company, TRW, Inc., U.S. Air Force Research Laboratory, U.S. Army Research Laboratory, US Army TACOM, Xerox Corporation.

We also thank the definers and developers of the first three versions of WinWin: Ellis Horowitz, Dan Port, Prasanta Bose, Yimin Bao, Anne Curran, Alex Egyed, Hoh In, Joo Lee, Jure Lee, Mingjune Lee, and Jungwon Park; and users of the four WinWin systems: Frank Beltz (TRW), Garry Brannum (Northrop Grumman), Walter Green (Aerospace), Elizabeth Kean (AFRL), Judy Kerner (Aerospace), Julie Kwan (USC), Andrew Landisman (TRW), Anne Lynch (USC), Ray Madachy (Litton), Azad Madni (Perceptronics), Nikunj Metha (USC and MediaConnex), Steve Mosher (USC), Karen Owens (Aerospace), Arnold Pittler (Motorola), and John Salasin (DARPA).

## 8 Sidebar: What is a Group Support System (GSS)

Some group tasks can be most effectively accomplished by carefully coordinated individual efforts. Technologies to support this kind of teamwork abound – e-mail, team calendaring, shared document repositories, and so on. Other tasks, like requirements negotiation, require concerted reasoning by many minds. For such tasks there are group support systems (GSS).

On the surface, a GSS might seem like a collection of glorified chat tools with some voting tools thrown in for good measure. For example, most GSS suites include shared list tools. Any user can make a contribution to a shared list at any time, and any contribution made by one person appears instantly on the screens of all other users. Various GSS suites include shared outlines, shared comment windows, shared drawing tools, and so on. In each tool, all the users can “talk” at once, contributing to the discussion as inspiration strikes, not having to wait for the floor. GSS suites usually include a variety of useful voting tools – Likert scales, semantic anchors, allocation votes, multi-criteria votes, and so on. The users can move their contributions into a vote tool, evaluate them, and then instantly review their results on line.

The real magic of a GSS though, is not what can be made to happen on the screen, but what can be made to happen in the group. Using a GSS, one can create predictable, repeatable patterns of human interaction and reasoning among people working toward a common goal. For example, most GSS include brainstorming tools that can be used to cause a group to diverge from their customary patterns of thinking, seeking farther and farther afield for new solutions. Some GSS also have idea organizing tools that can cause a group to structure the disorganized ideas running loose in their heads. Other GSS tools can cause a group to converge quickly from their many brainstorming ideas down to a clear focus on just the ideas that merit further attention.

Using a GSS, one can arrange a sequence of steps that a team can follow as they reason together to accomplish their task. In each step the GSS tools are configured so that as participants contribute to the system, a useful pattern of thinking will emerge.

In all, there are seven basic patterns of thinking a GSS can create in a group [Briggs, DeVreede, Nunamaker, & Tobey, 2001]:

- Diverge: Move from having fewer ideas to more ideas
- Converge: Move from having many ideas to a focus on just the few that are worthy of further attention
- Organize: Move from less understanding to more understanding of the relationships among ideas
- Elaborate: Move from having ideas expressed in less detail to having them expressed in more detail
- Abstract: Move from having ideas expressed in detail to having them expressed as fewer, more general concepts
- Evaluate: Move from less to more understanding of the value of concepts for accomplishing the task at hand.
- Build Consensus: Move from less to more understanding of the diverse interests of the group members; move from less agreement to more agreement about possible courses of action.

Besides requirements negotiation, GSS processes have been implemented for a variety of organizational tasks that require many people to think together. Here are a few examples:

- Strategic Planning
- New Product Development
- Marketing Focus Groups
- Total Quality Management
- Military Intelligence analysis
- Organizational Change Management
- Data modeling
- Group Therapy
- Factory Floor Design

Because a GSS operates over a computer network, it is often possible for team members to interact, even when its members are separated by oceans and continents. However, just as a screwdriver is not very useful for driving nails, a GSS is not right for every group interaction. Sometimes a team still needs to get together the old fashioned way, eyeball-to-eyeball, to see who sweats, and who blinks first. Nonetheless, extensive research shows that, under the right circumstances, teams using GSS can reduce their labor hours by 50% or more, and cut their project cycles by 60-90%. Such teams usually also report a higher-quality result than they were able to obtain using more conventional means [See Fjermestad, 2000 – 2001 for a compendium of field studies].