Analysis of System Requirements Negotiation Behavior Patterns

Alexander Egyed and Barry Boehm
University of Southern California
Los Angeles, CA 90089-0781 USA

Published in the Proceedings of 7th Annual International Symposium on Systems Engineering, 1997

ABSTRACT

Roughly 35 three-person teams played the roles of user, customer, and developer in negotiating the requirements of a library information system. Each team was provided with a suggested set of stakeholder goals and implementation options, but were encouraged to exercise creativity in expanding the stakeholder goals and in creating options for negotiating an eventually satisfactory set of system requirements.

The teams consisted of students in a first-year graduate course in software engineering at USC. They were provided with training in the Theory W (win-win) (Boehm-Ross, 1989) approach to system requirements determination and the associated USC WinWin groupware support system (Boehm, et al, 1995)(Horowitz, 1996). They were required to complete the assignment in two weeks.

Data was collected on the negotiation process and results, with 23 projects providing sufficiently complete and comparable data for analysis. A number of hypotheses were formulated about the results. Some of the hypotheses addressed the use and potential improvement of WinWin features. Others addressed more general process issues, e.g. that the uniform set of initial conditions would lead to uniform results. This paper summarizes the data analysis, which shows that expectations of uniform group behavior were generally not realized.

In terms of goals for relevant Capability Maturity Models (CMM's), this implies that the System Engineering CMM's Level 2 goal of Planned and Tracked processes is more realistic than the Software CMM's Level 2 goal of Repeatable processes.

KEYWORDS

Integrated Product Development, Teams, Case Studies, Software Systems Engineering, Requirements, Capability Maturity Models

INTRODUCTION

A major objective in several software process initiatives is to achieve repeatable processes, to which techniques such as statistical process control may be applied. For example, Level 2 of the SEI Software Capability Maturity Model (Paulk, et al, 1995) is called the Repeatable level. An early description of the model (Humphrey, 1989) states:

> *"Dr. W.E. Deming, in his work with the Japanese after World War II, applied the concepts of statistical process control to many of their industries. While there are important differences, these concepts are just as applicable to software as they are to producing consumer goods like cameras, television sets, or automobiles."*

On the other hand, level 2 of the Systems Engineering Capability Model (Bate, et al, 1995) is called the Planned and Tracked level. This indicates that expectations for systems engineering processes to be repeatable (i.e. for several systems engineering groups starting from the same problem statement to arrive at the same solution by the same sequence of steps) are not as high as those implied by the software CMM.

If one is performing systems engineering for a software-intensive-system, how repeatable should one expect the process and results to be? The software CMM would lead me to expect more repeatability than would the systems engineering CMM. It is very difficult to obtain comparable observational or experimental project data to explore this and related issues.

An opportunity arose to experiment on such issues in the context of a first-year graduate course on software engineering. In this course, student teams were trained in a particular system requirements approach (win-win). They then used the approach to negotiate a set of requirements for a Library Information System, for which a

common baseline problem description and candidate set of system functions were provided.

Given the prepackaged nature of this activity, one might expect that the teams would execute repeatable processes and produce repeatable results. Hypotheses were formulated and tested on several dimensions of this issue; the results are provided below.

Although the student-and-coursework context of this requirements activity is only moderately representative of actual practice, it can be argued that the results of such a pre-structured activity would serve as a lower bound on the repeatability of system and software requirements engineering processes and results in general project practice. A primary objective of the experiment and analysis is to explore the nature of such lower bounds.

Context

An attractive approach for determining system requirements is to use Integrated Product Teams (IPT's) composed of representative system stakeholders. However, there are few guidelines for the process by which the IPT should converge on a solution, and few criteria for determining when one has achieved a satisfactory solution.

We have been developing a systems management theory (Theory W), a systems engineering and development process (the Win-Win Spiral Model), and an IPT groupware support tool (WinWin) which provide such guidelines, criteria, and processes.

Theory W (Boehm-Ross, 1989) provides the principle and criterion that your project will succeed if and only if you make winners of all the critical stakeholders. Clearly, if all the stakeholders (users, customers, developers, maintainers, interoperators, etc.) are delighted with the project outcome, your project has succeeded. However, if you create a win-lose situation, the losing stakeholder can generally work out a way to survive at the expense of the other stakeholders, producing an unsuccessful project. For example, if a customer and user drive a developer to underbid a set of ambitious requirements, the developer can generally reinterpret the requirements in a way that is contractually compliant, but which leaves the customer and user with a low-value system.

The Theory-W concepts have been incorporated into the WinWin spiral process model (Boehm, et al, 1995) shown in Figure 1. The spiral model approach (Boehm, 1988) incorporates an evolutionary process which allows, and even encourages, multiple iterations of system and software development stages until the final goal is achieved. The Theory-W extension modifies the spiral model in such a way that each iteration identifies all critical stakeholders and their win conditions in advance.

As compared to hierarchical sequentialized systems engineering processes such as SPC's IDEF0-based Integrated Systems and Software Engineering Process (Rose, et al, 1996), the WinWin spiral model emphasizes cycles of concurrent elaboration. In each cycle, the key system definition artifacts (operational concept description, system requirements, system prototypes, and life cycle plan) are elaborated in a risk-driven fashion, along with a system rationale artifact which establishes the feasibility and consistency of the other artifacts.

The WinWin spiral model has also added the concept of "anchor points" to synchronize the spiral cycles with critical management decision points. The front-end anchor points are a Life Cycle Objectives (LCO) milestone and Life Cycle Architecture (LCA) milestone. (Boehm, 1996) provides more detail on how these milestones relate to spiral cycles in various situations. For system engineering and architecting of hardware-software systems (Rechtin-Maier, 1997) provides a similar synchronizing mechanism between software spiral cycles and intermediate hardware configurations, represented as intermediate circles in a spiral diagram.

To complement the process model, a group-

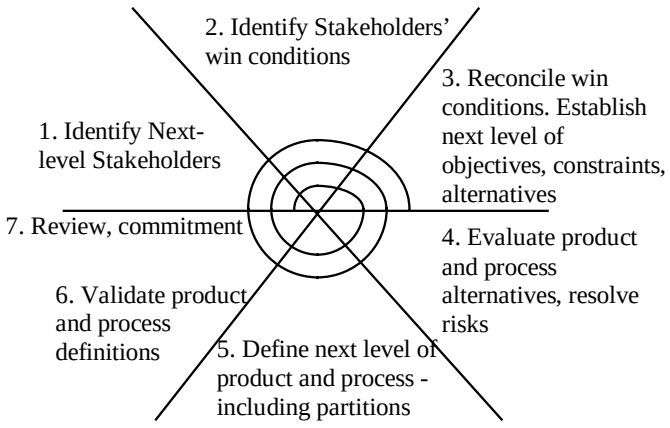

Figure 1. WinWin Spiral Model (Boehm, et al, 1994)

ware tool support system has been developed to ensure that the right amount of communication and collaboration is guaranteed during all cycles of the system development process. This tool is WinWin which is described below.

The WinWin Groupware Support System

WinWin is a groupware system which was primarily designed to be applicable for system and software requirements engineering. Stakeholders use the tool to identify win conditions, resulting conflicts, possible options, and finally solutions. The goal is to work towards agreements which incorporate all win conditions and resolve all conflicts. Additional negotiation aids such as domain taxonomy, glossary, and risk-resolution tools support the stakeholders in their efforts (Horowitz, 1996).

The WinWin negotiation model is primarily based on four artifact types: *Win Conditions*, *Issues*, *Options*, and *Agreements*. Win Conditions capture the stakeholders' goals and concerns with respect to a new system. If a Win Condition is non-controversial, it is adopted as an Agreement (see Figure 2). Otherwise, an Issue artifact is created to record the resulting conflict among Win Conditions. Options allow stakeholders to suggest alternative solutions which can resolve Issues. Finally, Agreements may be used to accept these solutions.

Each of the four artifact types contains information which is helpful for negotiations. Detailed information on the artifacts is available in an expanded Web version of this paper (Egyed-Boehm, 1997).

The WinWin model has been formally specified and analyzed for consistency (Lee, 1996) but only little is known about the correctness and usefulness of assumptions made during this process. Many questions have been raised such as these:

How are people and negotiation results affected by using a negotiation tool such as WinWin?

How similar are the negotiation results if stakeholders for all groups have a similar win conditions to start with and a pre-defined negotiation model to follow?

How do people factors, like work experience or age, effect the process and the outcome of the negotiation?

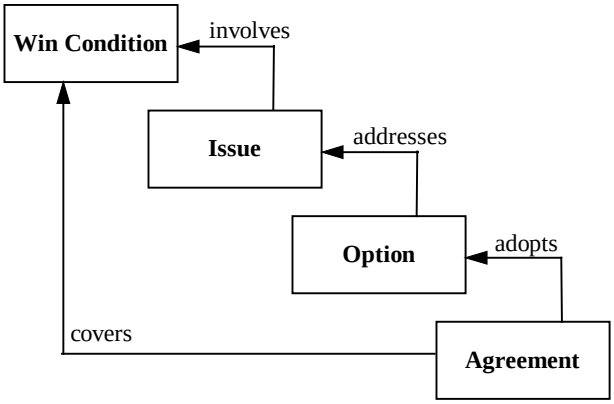

Figure 2. Inter-Artifact Relationship (Boehm, et al, 1995)

Do people use the tool as it was anticipated by the model?

And there are many more unanswered questions. Some of these questions are general; others are more related to the specific WinWin methodology. However, knowing the answers to these questions is vital in providing more useful and powerful negotiation aids for stakeholders.

THE PROJECT

The WinWin usage analysis was based on student projects. The projects are a key part of USC's core course in Software Engineering, which is designed to meet the increased needs for people capable of integrating systems and software engineering in an applications context (Boehm, 1994). The goal of these projects was to negotiate functionality, budget, architecture, and schedule for a proposed library system, called LSDI (Library Selective Dissemination of Information), to be used at the hypothetical SCU university (Southern California University). The university has three major campuses, each of which has a main library which will provide LSDI services. Each of the three campuses will operate a server running a COTS client-server library service package.

In addition, the SCU Library, Computer Science Department, and Computing Services Operation have been funded to develop an experimental *Selective Dissemination of Information* (SDI) system to provide SCU users with information about new library acquisitions of interest. It will do this by comparing attributes of new library acquisitions with interest profiles provided by the library users. The funding grant provided

initially $1,350K for development of the system and additional money for maintenance.

The basic components for the system were given prior to the negotiations. However, the students had the freedom to agree on different levels of detail for each component, or on whether it should be implemented or not. They could also add new components if desired. Details of the components are in the Web version (Egyed-Boehm, 1997).

Each level of each capability was characterized by its number of lines of code and other COCOMO cost drivers, enabling the stakeholders to use COCOMO for analysis of tradeoffs among cost, schedule, functionality, performance, and reliability.

Besides the software components, the students had to choose between two types of processors which would be used for the central library server. Processor X was a slower but reliable processor whereas processor Y was a faster but less mature processor. Additionally the size of the COTS integration was affected by the choice of the processor. The use of processor X required less lines-of-code integration for COTS than processor Y.

The budget was initially $1,350K but was changed later to $1,100K. No constraints were defined for the schedule but the different stakeholders had different requirements on how fast the system should be available.

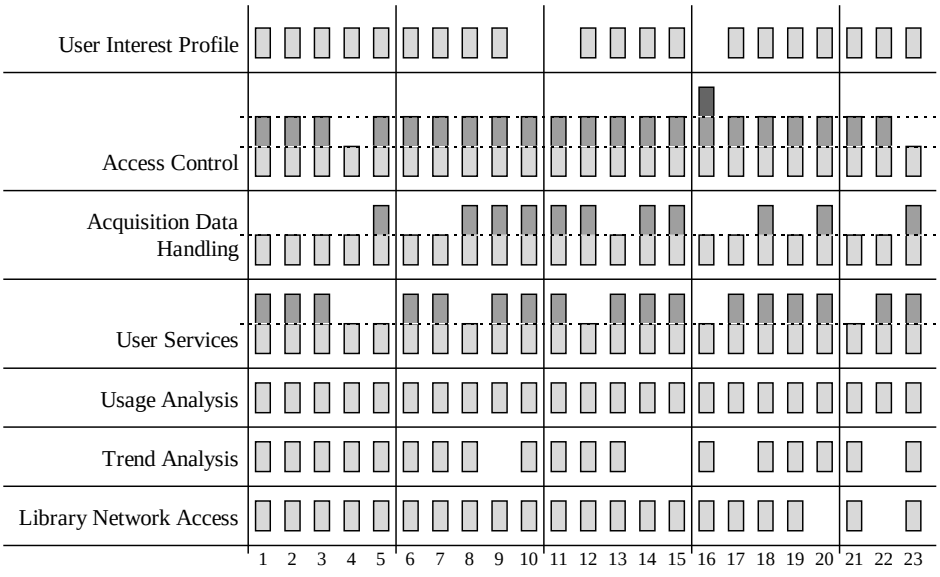

Figure 3. Functional Results

Schedule and People

Details on the negotiation schedule are in the Web version (Egyed-Boehm, 1997).

Limitations and Constraints

The analysis had to focus primarily on the end results, and only partially on the way people reached these results. This was caused because only the final team results were captured (the negotiation exercise was followed by a storage intensive architecture exercise which caused the students to delete their negotiation files before we had captured them; we have since developed a more robust instrumentation capability).

Another limitation was that it could not be assured that all teams worked independently. Interaction and information exchange might, therefore, have caused students to change their minds. However, based on observations during the short time frame in which the students had to do the requirements negotiation (only 2 weeks), it can be assumed that information exchange was not very extensive.

PROJECT ANALYSIS

A number of hypotheses were formulated about the uniformity or repeatability of the negotiation results across the 23 teams. The data analysis resulted in rejection of several of these hypotheses, indication that the results were not repeatable even in this prestructured situation. For some other hypotheses, however, the uniformity hypothesis was sustained by the data.

Functional and Non-Functional Requirements

Since functionality of the new system was the single most important negotiation factor, the outcome of it is especially interesting.

Because of the fact, that all students had a similar educational background, the same negotiation goals, and a pre-defined process model to follow, it was assumed that *most teams would come up with uniform functional results for the LSDI system.*

Figure 3 lists the functionality results of all teams. No box means that this function was not specified to be built. A light gray colored box means basic level, gray means extended level, and dark gray means rigorous level. The 23 groups produced 12 different results in functionality. The most common result (see e.g. group 1 in Figure 3) was used seven times. The next common ones were used three times, three times, and two times. All other teams negotiated different results.

A significant issue was the relative priority of non-functional requirements, such as performance and reliability. It is frequently hazarded that computer science students are more enamored with performance than reliability. Thus, another hypothesis was that *most teams would choose the faster processor Y instead of the slower, but more reliable processor X*. This turned out to be false because all teams chose to use processor X. Reliability was therefore rated far more important than performance, indicating that the student teams were doing a reasonable job of customer and developer role-playing.

Negotiation Cardinality

Additional hypotheses, based on the previous ones, were created which reflected assumptions about the negotiation processes. For all teams to come up with uniform solutions, it was assumed that the solution paths (processes) must have been uniform as well. A uniform solution path does not only imply a similar number of artifacts but also a similar way of connecting and commenting them.

One set of hypotheses was therefore that *the number of artifacts, connections, comments, etc. created by the stakeholders would be similar for all teams (±10%).* These hypotheses were clearly rejected as Tables II and III indicate. As indicated in line 3 of Table II, the hypothesis of uniform number of artifacts was rejected at the 19.2% level (a significance level of less than 5% is generally considered necessary for acceptance of a hypothesis). Lines 4-6 of Table II indicate that the uniformity hypotheses for numbers of connections, comments, and attachments were rejected between 57.5% and 75.3%.

Table II summarizes the experiment hypotheses and their level of significance. The smaller the number the stronger the support for this hypothesis (a level of significance below 5% indicates strong support). Table II highlights accepted hypotheses through gray shaded areas. Dark gray shaded areas indicate hypotheses with a level of significance below 1% (very strong support). These significance levels were used to accept or reject the hypotheses in this paper.

The diversity in outcome is not only reflected in the functionality but also in the way the students used the WinWin tool. The average, minimal, and maximal number of artifacts, connections, comments, and attachments show high ratios between the evaluated projects (1 to 4 for artifacts, 1 to 8 for connections). This is a clear indication that the project groups came up with very different ways of solving their problems.

To get more insight into the usage of artifacts, some hypotheses were formulated about the car-

Table I. Summary of WinWin Entries

	Average	Minimum	Maximum
Artifacts	46.1	21	80
Connections	65.5	23	188
Comments	19.5	0	101
Attachments	2.3	0	16

Table II. Level of Significance for Hypotheses

No.	Hypothesis	Level of Significance
1	Budgets are within 10% of $1,100K	<<1%
2	Schedules are within 10% of mean	<<1%
3	Number of artifacts are within 10%	19.2%
4	Number of connections are within 10%	57.5%
5	Number of comments are within 10%	69.4%
6	Number of attachments are within 10%	75.3%
7	Customers have more artifacts than users	<1%
8	Developers have more artifacts than users	<1%
9	Customers have more artifacts than developers	13.8%
10	More comments on Issues than on Win Conditions	42.7%
11	More comments on Issues than on Agreements	2.1%
12	More comments on Options than on Win Conditions	28.8%
13	More comments on Options than on Agreements	35%
14	More comments on win cond. than on Agreements	2.1%
15	Win Conditions were used longer than Issues	1.8%
16	Win conditions were used longer then Options	1.3%
17	Win Conditions were used longer than Agreements	2.4%
18	Issues were used longer then Options	47.4%
19	Agreements were used longer than Issues	30.4%
20	Agreements were used longer than Options	27.6%

dinality of the different artifact types (Win Conditions, Issues, Options, and Agreements). Two further hypotheses were that *Win Conditions would be the most common artifact type because they represent the knowledge base and that there would be more Options than Issues*. These hypotheses were accepted (see the level of significance of less than 1% for lines 21 and 22 in Table II).

Further details on the distribution of artifacts by stakeholder are in the Web version (Egyed-Boehm, 1997).

Cost and Schedule

The cost and schedule for the software development were calculated with the software cost estimation tool COCOMO 81 (Boehm, 1981). This tool derives the cost and schedule from the effort required to build the system. The effort is based on the estimated number of lines-per-code and a set of adjustable cost drivers, which allow the incorporation of software, project, and people factors. As discussed above, the teams were provided with size and cost driver ratings for each of the candidate Library SDI component choice levels. The cost estimations yielded by COCOMO are typically accurate within 20% of the actuals around 70% of the time.

A hypothesis was that *the teams' negotiated budget would be within 10% of available budget*. The budget of the library system was restricted to $1,100K for the final system. It turned out that all teams used up most of their money saving in average less than $50K (or $100K at most). One team used more money than actually allowed. Thus, no team saved more than 10% of its budget (see also level of significance for line 1 in Table II).

Similar to the cost results, it was assumed that *the development schedules would be very similar for all projects (±10%).* This turned out to be true for almost all projects. (see also level of significance of line 2 in Table II). Only one team decided to build the system faster by applying a low COCOMO schedule cost driver in their cost/schedule estimation. This cost driver incorporates the effects of building the system faster by using more developer and sacrificing, for instance, functionality to afford it. Most teams came up with a schedule of 15 months. Two teams needed more than 16 months. The team with the low schedule cost driver was able to develop the system within 13 months. Despite the desire of the user stakeholders to build the system as fast as possible, only one team finally agreed to do so.

Negotiation Interaction

The main interaction method in WinWin is via Comments. Comments can be attached to all types of artifacts (e.g. Win Conditions, Issues, Options) by all stakeholders. Additionally, Comments give stakeholders the ability to add their ideas to somebody else's artifact because only the owner of an artifact is allowed to change it's contents.

It was therefore interesting to know, whether students used the Comments extensively to do their negotiation or whether they performed face-to-face communication. It was assumed that *stakeholders would use primarily Comments and only secondary face-to-face communication*. Many groups chose to use Comments for their negotiation. However, a number of the students' post-project critiques indicated that they relied considerably more on face-to-face negotiation. This is consistent with the observation that only a few teams had a complete coverage of all necessary information (e.g. critics, rationale, etc.).

Furthermore, it was assumed that different artifact types have different needs for communication. The hypothesis was that *Issues and Option would be more controversial than the other artifact types*. Lines 10 though 14 in Table II show that different groups had indeed different views on where to use Comments. It appears that Comments were used far more frequently for Win Conditions, Issues, and Options rather than for Agreements.

Another issue investigated was whether the tool was used synchronously most of the time. Details on this issue are also in the Web version (Egyed-Boehm, 1997

SUMMARY

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 23 teams produced 12 different results in functionality. Repeatability was not achieved despite simplified project assumptions and a predefined negotiation process. Even those projects

which came up with the same results achieved them through different negotiation paths.

Reliability and better functionality of the software systems were far more important for the stakeholders than performance.

All teams chose to use the more mature and reliable processor X instead of the newer, faster, but less reliable processor Y. There was a possibility that some the student teams would be enticed by the speed and novelty of processor Y. But all the teams were responsive to the larger number of win conditions for reliability and functionality achievable via processor X.

Tool supported interaction provided through Comments could not replace face-to-face communication between the stakeholders.

The Comments method supported by WinWin should not be the only method to support group communication and collaboration but it is a helpful one. Many students used them, however, only very few teams used them extensively. Several students indicated that integrated use of e-mail would have been valuable. We are exploring this and other opinions such as audio and video attachments to artifacts and videoconferencing.

Most people tried to solve their conflicts on an equal level with similar participation. Only very few groups seemed to have had a strong leader dominate the creation of artifacts.

When it comes to negotiation, it is very important that all critical stakeholders have an equal opportunity to participate in the negotiation(s) and thus ensuring that everybody *will* become a winner (note: participation was 'measured' in number of artifacts and Comments used by stakeholders). This equal-participation pattern is probably characteristic of peer-to-peer negotiations (e.g. independent user, customer, developer), but might not hold across asymmetric power structures (e.g. boss-subordinate). Also, participation was not exactly equal, as discussed next.

Even though most stakeholders participated extensively during the negotiation, the users produced clearly fewer artifacts of all types than the other stakeholders.

The users had fewer artifacts than the customers or developers in almost all project groups regardless of artifact types. It should not be assumed that this phenomenon was caused because the user had fewer goals than the developer and customer. In reality the number of user Win Conditions were almost as high as the equivalent ones of the customer and the developer. The user produced far less Issues and Options, and these are activities were the users were equally qualified to participate since all stakeholder were computer science students in real life. The answer could be that the most significant Issue driver involved was meeting a limited budget, which affected the customers and developers more immediately than the users.

Most teams did not finish their negotiations in one session but required multiple sessions, negotiating for as long as 13 days.

Even though the problems the students had to solve were rather simple ones (many real life risks could be ignored or were simplified in the task statements), it seems that it was still very difficult to come up with agreements. This indicates that collaboration is a very difficult task which requires some training. This was borne out by comments in the students' post-project critiques, which indicated they felt they had learned a great deal about how to negotiate requirements.

People tend to accept the results of analytic tools (such as COCOMO) without questioning them.

The cost estimation tool COCOMO was used not only to decide on cost but also to come up with schedule and functionality. However, people tend to accept its results as 'given facts' even though its "accuracy within 20%, 70% of the time" was presented in class and in the model description. All teams used up most of their money ($1,100K) saving in average less than $50K (less than 5%) for risk contingencies.

CONCLUSIONS

The results based on these facts may not necessarily be applicable in all areas of group behavior because the team sizes were rather small and people factors, such as experience and age, were diverse (see also (Bullen-Bennett, 1990) for other observations). Conclusions derived from this analysis must, therefore, considered suggestive rather than definitive with respect to other situations.

Nevertheless, this experiment shows how differently the same problem may be solved by different people. A list of goals together with a gen-

eral process model on how to proceed does not achieve repeatability. Thus, for both systems engineering and software engineering processes, we conclude that the System Engineering CMM's Level 2 goal of Planned and Tracked processes is more realistic than the software CMM's goal of Repeatable processes.

ACKNOWLEDGMENTS

This research is sponsored by DARPA through Rome Laboratory under contract F30602-94-C-0195 and by the Affiliates of the USC Center for Software Engineering: Aerospace Corp., Air Force Cost Analysis Agency, AT&T, Bellcore, DISA, Electronic Data Systems, E-Systems, Hughes Aircraft, Interactive Development Environments, Institute for Defense Analysis, Jet Propulsion Laboratory, Litton Data Systems, Lockheed Martin, Loral Federal Systems, Motorola, Northrop Grumman, Rational Software, Rockwell International, Science Applications International, Software Engineering Institute, Software Rome Laboratory, US Army Research Laboratory, and Xerox.

Special thanks to Ming June Lee, Brad Clark, Cristina Gacek, and other students, faculty, and staff at the Computer Science Department, University of Southern California.

REFERENCES

Bate, R., et al, "A Systems Engineering Capability Maturity Model," Technical Report CMU/SEI-95-MM-003, Software Engineering Institute, Pittsburgh, PA 15213, November 1995

Boehm, B.W. *Software Engineering Economics*, Prentice Hall, 1981

Boehm, B.W. "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, May 1988, pp. 61-72

Boehm, B.W. and Ross, R. "Theory W Software Project Management: Principles and Examples," *IEEE Transactions on Software Engineering*, July 1989, pp.902-916

Boehm, B.W., Bose, P., Horowitz, E., Lee, M.J. "Software Requirements As Negotiated Win Conditions", *Proceedings of ICRE*, April 1994, pp.74-83

Boehm, B.W., "Integrated Software Engineering and System Engineering," *The Journal of NCOSE*, Vol. I, No. I, July/September 1994, pp. 61-67

Boehm, B.W., Bose, P., Horowitz, E., Lee, M.J. "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach", *Proceedings of ICSE-17*, April 1995, pp.243-253

Boehm, B.W., "Anchoring the Software Process," *IEEE Software*, July 1996, pp.73-82

Bullen, C.V., Bennet, J.L., "Learning from User Experience with Groupware", Conference on Computer-Supported Cooperative Work, October 1990, pp.291-302

Egyed, A., Boehm, B., "Analysis of System Requirement Negotiation Behavior Patterns (expanded version)," USC-CSE Technical Report, January 1997, at http://sunset.usc.edu/TechRpts/usccse97-506.html

Ellis, C.A., Gibbs, S.J., Rein, G.L., "Some Issues and Experiences", *Communications of the ACM*, Vol. 34, No.1, January 1991, pp.38-58

Horowitz, E. "WinWin Reference Manual: A System for Collaboration and Negotiation of Requirements", Center for Software Engineering, University of Southern California Technical Report, March 1996

Humphrey W.S., *Managing the Software Process*, Addison-Wesley, 1989

Lee, M.J., "Foundations of the WinWin Requirements Negotiation System," Ph.D. Dissertation, Center for Software Engineering, University of Southern California Technical Report, May 1996

Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B., *The Capability Maturity Model - Guidelines for Improving the Software Process*, Addison-Wesley, 1995

Rechtin, E., Maier, M., *The Art of Systems Architecting*, CRC Press, 1997

Rose, S., et al, "Integrated Systems and Software Engineering Process," Software Productivity Consortium Report SPC-96001-CMC, Herndon, VA, May 1996

AUTHOR BIOGRAPHIES

Alexander Egyed is a PhD student at the Center for Software Engineering at USC. His research interests are in software architecture, software processes, and requirements negotiation. He received a Dipl.-Ing. in computer science from the Johannes Kepler University of Linz, Austria and a MS in computer science from USC.

Barry Boehm is the TRW Professor of Software Engineering and Director of the Center for Software Engineering at the University of Southern California. Boehm received a BA in mathematics from Harvard University and an MS and PhD in mathematics from the University of California at Los Angeles. He is a fellow of the IEEE and the AIAA.

You may reach the authors at:

+1 213 740 5703
{aegyed, boehm}@sunset.usc.edu
http://sunset.usc.edu/