Improving the Life-Cycle Process in Software Engineering Education

Barry Boehm and Alexander Egyed[1])

Published in the Proceedings of the European Software Day (as part of the 24th Euromicro conference), 1998

Abstract
The success of software projects and the resulting software products are highly dependent on the initial stages of the life-cycle process – the inception and elaboration stages. The most critical success factors in improving the outcome of software projects have often been identified as being the requirements negotiation and the initial architecting and planing of the software system.

Not surprisingly, this area has thus received strong attention in the research community. It has, however, been hard to validate the effectiveness and feasibility of new or improved concepts because they are often only shown to work in a simplified and hypothesized project environment. Industry, on the other hand, has been cautious in adopting unproven ideas. This has led to a form of deadlock between those parties.

In the last two years, we had had the opportunity to observe dozens of software development teams in planing, specifying and building library related, real-world applications. This environment provided us with a unique way of introducing, validating and improving the life cycle process with new principles such as the WinWin approach to software development. This paper summarizes the lessons we have learned.

1. Introduction

1.1. The People

In 1996, 15 development teams used the WinWin spiral model approach to develop multimedia-related library applications for the Library of the University of Southern California (USC). The development teams consisted of an averaged of six USC graduate students (Master and PhD students) per team with a mix of about 70% fresh (only little experienced) students and 30% experienced practitioners from industry (the latter were mostly working for companies and attended the course via the instructional television network).

---

[1] University of Southern California, Center for Software Engineering, Salvatori Building, Los Angeles, CA 90089-0781.

In 1997, 16 development teams used an improved version of the same WinWin approach to again develop library related applications for the USC Library. This time the domain of applications was broader, covering also the administrative side of Library operations. The students teams averaged five per team and the ratio of experienced vs. inexperienced students was similar.

In both years there were almost as many projects as there were students teams. In a few cases, more than one team worked on the same problem set. The problem sets were provided by librarians from the USC Library and their problem descriptions were initially nothing more than 1-2 paragraph statements (see Table 1 for a list of most projects).

The students were supervised by faculty and staff of the USC Center for Software Engineering who also taught them software engineering principles and provided technical support. Further outside support was provided by the University Computing Services (UCS).

1.2. The Projects

The projects were developed over a period of two semester. Thus, the 1996 projects started off in Fall 1996 and continued through the spring semester of 1998 (or in a few cases until the end of Summer 1998). Similarly, the 1997 projects started in Fall 1997 and went through Spring and in one case Summer of 1998.

Unfortunately, the attendance of the Fall and Spring classes are not the same. The Fall course is part of the core requirements for the USC graduate program in computer science – not so, the Spring course. This usually leaves us with only a third of the students to continue the projects in the second semester. Thus, we are forced to abandon some of the projects in mid-time and continue only some of them. Table 1 shows most of the projects of both years. The projects marked with (1) are projects we continued for the second semester. In rare cases, where we find that the projects seem to be heading to a similar solution, we merge them together and both are continued for a second semester (2).

1.2. Real-World Characteristics

The personnel strain, however, also adds to the realism of the projects. In fact, the projects exhibited a number of real-world characteristics as listed below:

- Real customers and users, and thus real problems and conflicts to solve
- Fuzzy requirements
- Resource conflicts (availability and accessibility of hardware and software.

- Personnel conflicts (new people with new ideas join the teams; other people leave the teams)
- Solution needs to be integrated into existing USC Library operation. Independent ‘islands’ of solutions are not effective.

Table 1 - Some of the 1996 and 1997 Projects

1996-1997 Projects	**1997-1998 Projects**
Cinema-TV Moving Images[1]	Architecture & Fine Arts Databases
EDGAR Corporate Data	Bella Lewitsky Archives
Hancock Image Archive	Business School Working Papers[2]
Interactive TV Material	Inter-Library Loan[1]
Korean-American Museum	Engineering Technical Reports[2]
Latin American Pamphlets[1]	General Library FAQ's
Digital Maps	Hancock Museum Virtual Tour[1]
Medieval Manuscripts[1]	Lion Feuchtwanger Archive
Planning Documents[2]	Network Consultation Support
Searchable Archives for Images[2]	Serial Publication[1]
Stereoscopic Slides[2]	Statistical Charts[1]
Technical Reports[1]	Virtual Education Reference Assistant

Most of the projects, so far have been considered a high success for our customer, the USC Library. Not only did they commit to continue the projects for a second year but also preparations for the third year have been set in motion. This continued alliance between the Center for Software Engineering and the USC Library proves to be a win-win not only for the Library but also all other parties involved as Table 2 shows:

Table 2 - Stakeholder Win-Win Approach

Stakeholders	**Win conditions**
Developers (Students)	• Full range of software engineering skills • Real-client project experience • Advanced software technology experience
Customers (Librarians)	• Useful applications • Advanced software technology understanding • Moderate time requirements
Faculty and Staff	• Educate future software engineering leaders • Better software engineering technology • Applied on real-client projects

1.2. Outline

Experiences with this project have been summarized in [7]. This paper concentrates on how the projects helped us in validating and improving software engineering technology.

In the following section we will summarize the software process we used in the first year. Following that, we will discuss the improvements for the second year based on the lessons we learned from the previous one. We will conclude this paper by summarizing some of our future goals.

2. First Year Development

The first year, we used a number of models to help develop the projects. The most important ones were the WinWin Spiral Model [6], the WinWin Negotiation Model [5], and COCOMO [2]. It is out of the scope of this paper to address them in detail. We will therefore only concentrate on the first one.

2.1. Process Models

The WinWin spiral model is a derivative of the original spiral model [3] which uses a cyclic approach to develop increasingly detailed elaborations of a software system. The original model focused on the following aspects:

- 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.

The Spiral Model has been extensively elaborated (e.g. [15]), and successfully applied in numerous projects (e.g., [13], [10]). However, some common difficulties have led to some further extensions to the model.

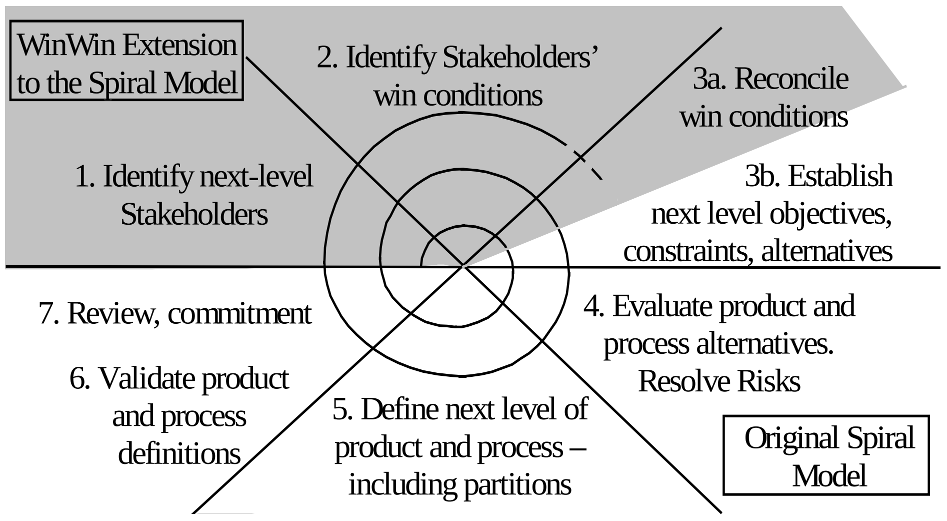
WinWin Extension to the Spiral Model
2. Identify Stakeholders' win conditions
3a. Reconcile win conditions
1. Identify next-level Stakeholders
3b. Establish next level objectives, constraints, alternatives
7. Review, commitment
4. Evaluate product and process alternatives. Resolve Risks
6. Validate product and process definitions
5. Define next level of product and process – including partitions
Original Spiral Model

Figure 1 - The WinWin Spiral Model

One difficulty involves answering the question, "Where do the elaborated objectives, constraints, and alternatives come from?" The WinWin Spiral Model resolves this difficulty by adding three activities to the front of each spiral cycle, as illustrated in Figure 1 [4].

- Identify the system or subsystem's key stakeholders.
- Identify the stakeholders' win conditions for the system or subsystem.
- Negotiate win-win reconciliation of the stakeholders' win conditions.

We found that these steps indeed produced the key product and process objectives, constraints, and alternatives for the next version [5]. The overall stakeholder WinWin negotiation approach is similar to other team approaches [7] but 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.

2.2. Process Anchor Points

The teams also followed the Anchor points described in [6]. There, two generally applicable milestones were defined for the WinWin spiral model, called the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) (see Table 3). Each milestone corresponds to one spiral cycle and the LCA milestones is a refinement (a later cycle) of the LCO. Each milestone is divided into milestone elements, such as operational concept, system requirements, software architecture, plan, and feasibility rationale. The table entries contain information of what is expected to be completed for a certain milestone and for a particular milestone element.

Table 3 - Contents of LCO and LCA Milestones

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
Definition of Operational Concept	Top-level system objectives and scope System boundary - Environment parameters and assumptions - Evolution parameters Operational concept - Operations and maintenance scenarios and parameters - Organizational life-cycle responsibilities	Elaboration of system objectives and scope by increment Elaboration of operational concept by increment
Definition of System Requirements	Top-level functions, interfaces, quality attribute levels, including: - Growth vectors - Priorities Stakeholders' concurrence on essentials	Elaboration of functions, interfaces, quality attributes by increment - Identification of TBDs (to-be-determined) Stakeholders' concurrence on their priority concerns
Definition of System and Software Architecture	Top-level definition of at least one feasible architecture - Physical and logical elements and relationships - Choices of COTS and reusable software elements Identification of infeasible architecture options	Choice of architecture and elaboration by increment - Physical and logical components, connectors, configurations, constraints - COTS, reuse choices - Domain-architecture and architectural style choices Architecture evolution parameters
Definition of Life-Cycle Plan	Identification of life-cycle stakeholders - Users, customers, developers, maintainers, interoperators, general public, others Identification of life-cycle process model - Top-level stages, increments Top-level WWWWWHH* by stage	Elaboration of WWWWWHH* for Initial Operational Capability (IOC) - Partial elaboration, identification of key TBDs for later increments
Feasibility Rationale	Assurance of consistency among elements above - Via analysis, measurement, prototyping, simulation, etc. - Business case analysis for requirements, feasible architectures	Assurance of consistency among elements above All major risks resolved or covered by risk management plan

* WWWWWHH: Why, What, When, Who, Where, How, How Much

An initial milestone, completion of the WinWin requirements negotiation was due at the end of Week 4. The LCO milestone was due in Week 6 and the LCA milestone was completed in Week 11 at the end of the first semester, including a prototype, which was mostly done as part of the second cycle. Thus, the net result of the first semester's activity was to go from the one-paragraph problem statements to LCA packages of roughly 200 pages plus a prototype.

The second semester started off by revisiting the LCA deliverables and continuing on to the IOC (Initial Operational Capabilities) milestone, which was due at the end of the second term. The ICO milestone is about:

- *User, operator and maintainer preparation*, including selection, teambuilding, training and other qualification for familiarization usage, operations, or maintenance

- *Software preparation*, including both operational and support software with appropriate commentary and documentation; data preparation or conversion; the necessary licenses and rights for COTS and reused software, and appropriate operational readiness testing
- *Site preparation*, including facilities, equipment, supplies, and COTS vendor support arrangements

More usage information of the anchor points as well as their entry and exit criteria are described in [7].

2.1. Gathered Data

Since we wanted to analyze how students would use our models and in what places they would encounter problems while applying them, we gathered extensive data throughout the development life cycle. The following summarizes some of the data we have:

- *WinWin Negotiation Tool:* Based on the WinWin negotiation model, it was designed to keep track of the changes of the negotiation. Besides the model implicit information the tool also captured other usage activities in detail.
- *Documentation:* Each LCO and LCA milestone element described above resulted in a document tailored towards it. The LCO package averaged in about 160 pages and the LCA package averaged in 230 pages.
- *Architecture Review Board* [1]: At the end of the LCA milestone, teams presented their solution and approach (architecture, etc.) to us and their clients. This provided us with some insights into team activities. However, we were not able to capture those quantitative.
- *Customer Questionnaires:* At the end of each semester (LCA and IOC milestones) we asked the USC Library customers and users to provide feedback to us in form of a questionaire. They were asked to summarize their experiences working with the students and whether or not the product satisfied their needs.
- *Student Critiques:* Similarly, at the end of the LCA and IOC milestones, we asked the students to summarize their experiences.
- *Weekly Metrics:* Students were asked to submit effort data on a weekly basis. The metrics forms they had to fill out were a matrix of the weekdays and major development activities.
- *COCOMO related questionnaires:* To further analyze the projects, the students had to fill out more detailed questionnaires about factors affecting development cost and effort.

3. Improving the Process and its Deliverables for Year Two

Using the information we gathered during the first year, we found a number of places where the process was not efficient enough to meet stakeholders’ expectations. Those deficiencies were:

3.1. Prototyping

In 1996, the development teams were required to produce a prototype at the end of the first semester and then a final product with sufficient initial capabilities at the end of the second semester (which for most team members also mark the end of their involvement in the development process).

Having had a prototype to show to the Library clients before actual construction initiated was highly beneficial. Although the librarians created the problem statement and participated in the requirements negotiation with the student teams and with various stages of the prototype, the final prototype presentations yielded insightful surprises. This had, however, also the downside that in the middle of the project life-cycle, the clients expectation of what is possible expanded – resulting in requirements changes late in the process.

To cope with this challenge, we decided to incorporate prototypes as early on as possible. In the second year, we followed the spiral model process more closely and produced prototypes as part of every cycle. The first one was build parallel to the requirements negotiation which incorportated user interface features. The second and third prototype were build with the LCO and LCA milestones respectively. This change was also reflected in Table 3 which got extended by the following milestone element:

System Prototype(s):

LCO:	LCA:
• Exercise key usage scenarios	• Exercise range of usage scenarios
• Resolve critical risks	• Resolve major outstanding risks

3.2. Architecture Review Board (ARB)

The end of the first semester (LCA milestone) featured also an activity that was similar to an Architecture Review Board meeting. The main participants were faculty and staff from USC Center for Software Engineering, USC Library clients, and their respective student teams.

Like with prototypes, all involved parties gained important insight into the development process and product, and how they would affect operation. To increase the benefit of those sessions, we introduced the ARBs to both the LCO and the LCA milestones – thus enabling the teams to incorporate stakeholder feedback earlier on. We further structured those meetings stronger since the first year's ones were rather informal.

3.3. OO Analysis and Design

In 1996, we primarily concentrated on providing process support for the high level activities such as requirements negotiation, LCO and LCA package creation. We left it, however, unspecified what design process the teams should follow.

The students turned out to be very resourceful in dealing with that, however, this situation was far from ideal when it came to analyzing what they did. We found it hard to reconcile their approach to extract best practices and pitfalls.

In the second year, we went to a more concise and integrated set of design views, based on the Unified Modeling Language (UML) and the Rational Rose toolset [9].

Using UML, the teams were able to more strongly refine their software architectural description and using a design tool turned out to be a great win-win for both the designers and the analyzers. The designs used more uniform methodologies which made it easier to communicate with teams (e.g. during ARBs) and the design model can now be more uniformly analyzed (e.g. we are currently attempting to develop a software sizing method based on UML).

3.4. Training

Another major problem in the first year was the issue of training the teams in using the spiral model, UML, and other models. We found that without adequate training the teams would fail to use the models and corresponding tools very effectively.

Since it is not feasible for the graduate program to add a prerequisite course in software engineering models (even though this would have been ideal), we decided to spend more time in the beginning of the first semester teaching those models. Fortunately for us, not all models are needed right from the beginning.

The additional training sessions in the usage of WinWin, Rose, COCOMO, and other tools turned out to be highly effective. With that, they had at least some tool experience before they used them in the projects.

3.5. Transition of Product

In the beginning, the library clients were considerably uncertain about going forward with the projects. This changed however soon after they saw the first prototypes. Nevertheless, in the first year we learned that most of the clients were not empowered to support the product not just with knowledge and enthusiasm, but also with resources to support the product's operation and maintenance. Most of the products we delivered did not see operation in the USC Library.

In the second-year projects, the transition of the projects became our top criterion for selecting projects. From the five projects we constructed in the second semester (of the second year), three are now transitioning to library operations, and the other two have good prospects for transition after refinement this summer.

3.6. Documentation

The first year, we structured the teams around the main deliverables of the LCO and LCA milestones, the documentation. We found, however, that this had a major risk associated with it – inconsistency. If each person in the team is given primary responsibility in creating one document then the team members must spend considerable time talking to each other to make sure their documents are consistent.

Table 4 - Project Characterisitcs

Project Characteristics	1996-1997	1997-1998
Architecture Teams	15	16
Applications Architected	12	15
Applications Developed	6	5
Applications Transitioned	1	3-5
LCO Spec Pages	160	110
LCA Spec Pages	230	154
Application Types	Multimedia	Multimedia, Text Archives, Ref. Service, Infrastructure

We found the concept of primary responsibility to work well enough to continue it in the second year. However, we realized that we had to ensure that the conceptual integrity of the documentation is maintained in the process. This was one of the reasons why we introduced ARBs during the LCO and LCA. However, we also found that the documentation guidelines we provided were redundant, causing unnecessary efforts in trying to keep them consistent. We therefore restructured the document guidelines to reduce duplication, and also to adapt them for use with UML. The result can be seen in Table 4. We successfully reduced the specification by an average of 40-50%.

3.6. Data Gathering

Improving the process, as it was described in the items above, was done to a good part so that additional or more precise metrics could be gathered throughout the second development life-cycle. The following describes the improvements in the metrics gathering process:

- Model and tool support was available for many life-cycle activities. Thus, information about their usage were captured (e.g. Rose)
- Weekly effort metrics were also gathered in the first semester (we had previously only gathered them during construction)
- COTS related questionnaires to analyze the cost and effort impact of Commercial-Of-The-Shelf (COTS) products were added. This made it possible since many teams were incorporated COTS products into their designs (e.g. the USC Library information system – SIRSI).
- Better structured student critiques and customer questionnaires. Looking at first years questionnaires we found that there were some issues we would have liked to have feedback from all clients and student.

4. Ongoing Improvements

The collaboration with the Library continues to be very beneficial for us in evaluating new ideas. We are currently involved in improving the process further. To some degree, we need to adapt and refine our existing process described above. However, we also would like our model to conform even stronger to industrial standards. The major challenges as we see them are listed below:

- Combining the Best of Standards: Our original standards were closely tailored to DoD standards (e.g. DoD 2167). We are now involved in integrating new standards such as the IEEE/EIA 12207.1-1997 [11] into our process model.
- COCOMO II: Our center created a new cost model called COCOMO II. For the next year, we hope to provide a version, tailored to our library domain, to our teams.
- The discontinuity in personnel between the two semesters continues to be a problem. We are investigation ways to ease that transition.

- Integrating the Rational UML/Objectory process [12] and the Unified Software Management [14] into our model/process.

We also continue to refine our own models. The most significant result has been the consolidation of many of our models into an integrated conceptual model called MBASE (Model-Based Architecting and Software Engineering) [8].

The MBASE model concentrates on avoidance of model clashes between process model, success model, product model, and property model. Figure 2 shows an excerpt of the process on how to get to the LCO milestone starting from the domain description and describing the dependencies of intermediate process stages. For instance, the WinWin negotiation model needs a WinWin Taxonomy and Stakeholders' win conditions as input and delivers the WinWin Agreements as an output.

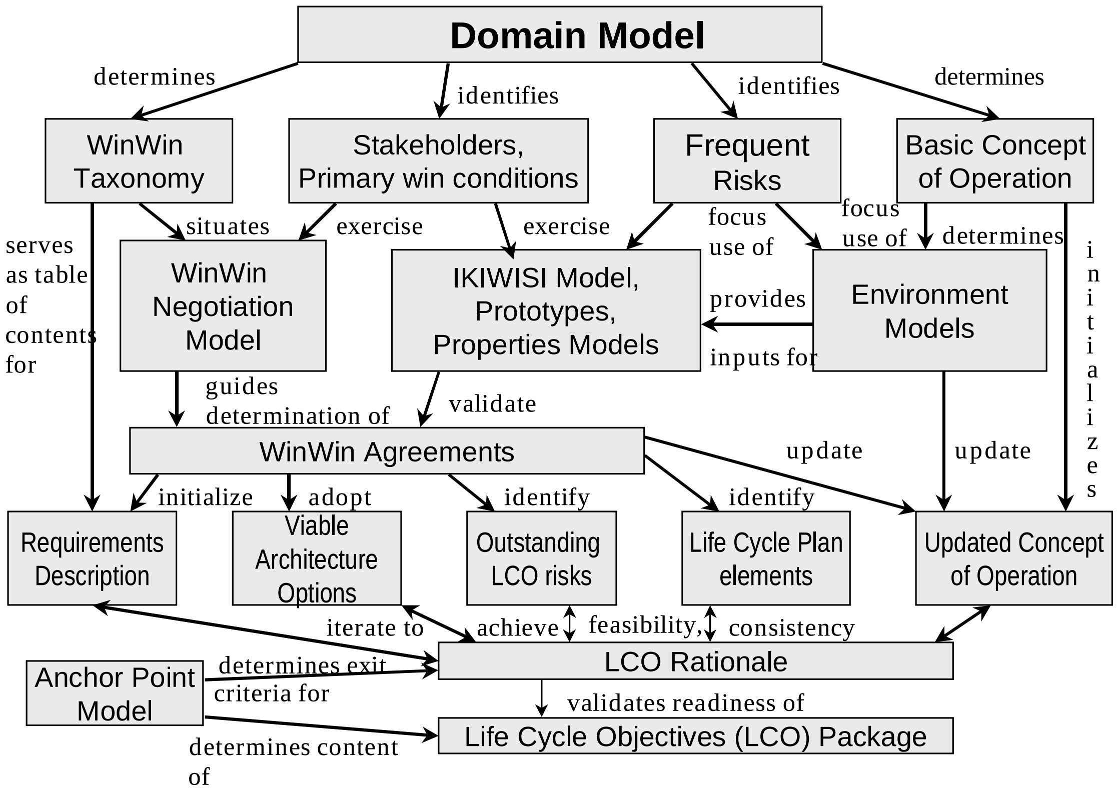

Figure 2 - Detailed Process for LCO Stage

The library project were and will continue to be an testing field for this kind of improvements. We are further collaborating with other Universities such as George Mason University (USA) and Johannes Kepler University (Austria) who have adopted some of our concepts. This will enable us to learn more about how our process works in other domains.

5. Acknowledgement

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, Allied Signal, Bellcore, Boeing, Electronic Data Systems, E-Systems, GDE Systems, Hughes Aircraft, Interactive Development Environments, Institute for Defense Analysis, Jet Propulsion Laboratory, Litton Data Systems, Lockheed Martin, Loral Federal Systems, MCC, Motorola, Network Programs, Northrop Grumman, Rational Software, Science Applications International, Software Engineering Institute, Software Productivity Consortium, Sun Microsystems, TI, TRW, USAF, Rome Laboratory, US Army Research Laboratory, and Xerox. We also thank Marwan Abi-Antoun, Julie Kwan, Ray Madachy, Dan Port, and Archita Shah for support in key areas.

4. References

[1] AT&T (1993), Best Current Practices: Software Architecture Validation, AT&T, Murray Hill, NJ.

[2] Boehm, B.W. (1981), *Software Engineering Economics*, Prentice Hall.

[3] Boehm, B.W. (1988), "A Spiral Model of Software Development and Enhancement," *Computer*, May, v. 21 no. 5, pp. 61-72.

[4] Boehm, B.W., Bose, P. (1994), "A Collaborative Spiral Software Process Model Based on Theory W," *Proceedings, 3rd International Conference on the Software Process, Applying the Software Process*, IEEE, Reston, VA, October.

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

[6] Boehm, B.W. (1996), "Anchoring the Software Process," *IEEE Software*, July, v.13 no.4, pp.73-82.

[7] Boehm, B.W., Egyed, A. F., Kwan, J., Madachy, R, Port, D., and Shah, A. (1998), "Using the WinWin Spiral Model: A Case Study," *IEEE Computer*, July (to appear).

[8] Boehm, B.W., Port, D. (1998), "Conceptual Modeling Challenges for Model-Based Architecting and Software Engineering (MBASE)," *Proceedings, Symposium on Conceptual Modeling*, Springer Verlag, (to appear).

[9] Booch, G., Jacobson, I., and Rumbaugh, J. (1997), "The Unified Modeling Language for Object-Oriented Development," Documentation set, version 1.0, Rational Software Corporation.

[10] Frazier, T. P., Bailey, J.W. (1996), "The costs and benefits of domain-oriented software reuse: Evidence from the STARS demonstration projects," IDA Paper P-3191, Institute for Defense Analysis.

[11] IEEE (1998), "Industry Implementation of International Standard ISO/EIC 12207: 1995", IEEE/EIA 12207.1-1997, April.

[12] Rational Software Corp. (1997), "Rational Objectory Process," Version 4.1, Rational Software Corp., Santa Clara, CA.

[13] Royce, W.E. (1990), "TRW's Ada Process Model for Incremental Development of Large Software Systems," *Proceedings, ICSE 12*, IEEE/ACM, March, pp. 2-11.

[14] Royce, W. E. (1998), *Unified Software Management*, Addison-Wesley, Reading, Mass. (to be published).

[15] Software Productivity Consortium (1994), "Process Engineering with the Evolutionary Spiral Process Model," SPC-93098-CMC, version 01.00.06, Herndon, Virginia.