ELSEVIER

Computer Standards & Interfaces 21 (1999) 63–75

COMPUTER STANDARDS & INTERFACES

# Optimizing software product integrity through life-cycle process integration

Barry Boehm [1], Alexander Egyed *

*University of Southern California, Center for Software Engineering, Salvatori Computer Science Building 328, Los Angeles, CA 90089-0781, USA*

Received 18 December 1998; accepted 25 January 1999

**Abstract**

Managed and optimized—these are the names for the levels 4 and 5 of the Capability Maturity Model (CMM) respectively. With that the Software Engineering Institute (SEI) pays tribute to the fact that, after the process has been defined, higher process maturity, and with that higher product maturity, can only be achieved by improving and optimizing the life-cycle process itself. In the last three years, we had had the opportunity to observe more than 50 software development teams in planning, 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 in our ongoing endeavor to integrate the WinWin life-cycle process. In doing so, we will not only describe what techniques have proven to be useful in getting the developer's task done but the reader will also get some insight on how to tackle process improvement itself. As more and more companies are reaching CMM levels two or higher this task, of managing and optimizing the process, becomes increasingly important. © 1999 Elsevier Science B.V. All rights reserved.

*Keywords:* Software product integrity; CMM; Process integration; WinWin; Spiral model; MBASE; View integration

## 1. Introduction

The major objective of the Capability Maturity Model (CMM) [18] in several software process initiatives is to achieve defined, managed, and optimized processes with the hope that this improvement in process maturity will also yield a higher maturity of the software (system) product.

The CMM does, however, not elaborate on what approaches and techniques are useful in achieving this goal. Especially the process optimization area, which comes rather late in the maturity hierarchy (in level 5), is an issue which has not been tackled strongly by the research or practitioner communities in general. Nevertheless, during the optimization level it is expected that ''the organization has the means to identify weaknesses and strengthen the process proactively, with the goal of preventing the occurrence of defects'' [18].

This case study will present the findings of the continuing process of observing over 50 develop-

* Corresponding author. Tel.: +1-213-7406504; E-mail: aegyed@sunset.usc.edu

[1] E-mail: boehm@sunset.usc.edu

ment teams over a period of 3 years—roughly 15 teams each year. Those teams, with an average team size of five people per team (as little as four and as high as seven people), were primarily involved in developing multimedia applications for the University of Southern California (USC) with the USC Library Services as their main customer. The user communities for these applications are for the most part faculty and students at the university and Library administrators and service providers. Since many of these applications were developed for the World-Wide-Web, the user community may be quite extensive in some cases.

Those USC Digital Library projects constitute a major proof of the success and feasibility of the WinWin Development Model (a process model consisting of the WinWin Negotiation Model and the WinWin Spiral Model) in that the projects were highly successful in their approaches and results. The developers in question were computer science students at USC, taking a graduate level course in Software Engineering. On average, 30% of the graduate students taking this class (and the successive ones) are experienced software developers from industry, who often take these courses remotely while they are continuing to work at their respective companies. This, fact also adds another interesting dimension to our findings—that of how more experienced teams, or teams with mixed experiences, use our process model differently from less experienced ones.

This collaboration between us, the Center for Software Engineering, and the USC Library is resulting in numerous advantages for all involved parties. The students taking our software engineering course gain valuable knowledge and skills in dealing with real-world software development challenges as a team, which poses greater challenges than the usual individual programming assignment. The USC Library in turn gains well-engineered software products for a very reasonable price (almost free) and with minimal disruption to the library operations. Furthermore, the industry (and especially our affiliates) benefit by having access to a better educated pool of future employees who have gained relevant skills in handling the entire software life-cycle. And finally, we (our center and the software engineering community in general) gain invaluable experiences in improving software engineering tools and techniques, and in validating them.

In improving this course we applied the same process model the development teams used in building their Library applications, the WinWin Spiral Model [7] (explained later), and combined it with Basili's Experience Factory [2]. This approach is summarized in Fig. 1 (see also Ref. [10]). We used the WinWin Spiral Model to develop the initial version of the course and its instrumentation. Each project also uses the WinWin Spiral Model to define, develop, and transition their respective application products. We then analyze the course instrumentation results, student critiques, client evaluations, and grading information to determine improvements for the course in the following year, using the Experience Factory paradigm. Since the development projects involve dealing with COTS products (Commercial-off-the-Shelf), architectural description languages (UML), and Web technology we also had to anticipate and adapt changes in those technologies.

### 1.1. The people

The first set of projects commenced in 1996, when 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 average 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

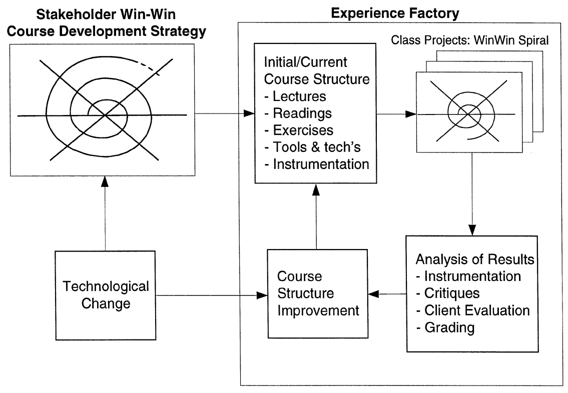


Fig. 1. Course development and evolution strategy.

and attended the course via the instructional television network).

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.

At the time of the writing of this article, the next rounds of projects have been initiated. In Fall 1998, we have 20 development teams working on 18 different applications. The requirements negotiation, and the first two iterations of the major deliverables (LCO and LCA milestones, see below) have been completed.

During all three years there were almost as many projects as there were student 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 one to two 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 Information Services Division (ICS) of USC.

### 1.2. The projects

The projects were (and are being) developed over a period of two semesters. Thus, the 1996 projects started off in Fall 1996 and continued through the spring semester of 1997 (or in a few cases until the end of Summer 1997). Similarly, the 1997 projects started in Fall 1997 and went through Spring and in one case through Summer of 1998. The first stage of the 1998 projects is about to be finished and the detailed design and implementation will follow in Spring. We expect further collaboration for the foreseeable future.

Since those projects are often quite extensive, the development is impossible to be compressed onto one semester only. To deal with this, we offer two consecutive software engineering courses where the

Table 1
Digital library projects

| 1996–1997 Projects | 1997–1998 Projects | 1998–1999 Projects |
|---|---|---|
| Cinema-TV Moving Images[a] | Architecture and Fine Arts Databases | Data Mining the Library Catalog |
| EDGAR Corporate Data[a] | Bella Lewitsky Archives | California Virtual University Database |
| Hancock Image Archive[b] | Business School Working Papers[b] | Dissertations |
| Interactive TV Material | Inter-Library Loan[a] | Virtual Education Reference Assistant |
| Korean-American Museum | Engineering Technical Reports[b] | Business Q&As |
| Latin American Pamphlets[a] | General Library FAQ’s | Asian Film Database on the Internet |
| Digital Maps | Hancock Museum Virtual Tour[a] | WWI—Record Enhancement/TOC |
| Medieval Manuscripts[a] | Lion Feuchtwanger Archive | Digital Document Creation and Storage |
| Planning Documents | Network Consultation Support | Current Awareness Service for Social Work Doctoral Students |
| Searchable Archives for Images[b] | Serial Publication[a] | Seaver Auditorium Scheduling |
| Stereoscopic Slides[b] | Statistical Charts[a] | Hispanic Digital Archive |
| Technical Reports[a] | Virtual Education Reference Assistant | Book Locator for Doheny Stacks |
| | | New Booklist |
| | | ARIEL to Web Document Delivery |
| | | Metadata Creation for Digital Records |
| | | Authoring tool for the ADE system |
| | | Voice Input for Metadata |
| | | Qualcomm Dinner Scheduling |

[a]Continued during the spring semester.
[b]Merged together to one project during the spring semester.

second one continues the projects from the first one in order to implement and transition them. 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-year and continue only some of them. Table 1 shows most of the projects of all three 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). As of this time we do not know which projects will be continued this year. We will determine this based on what students return (we favor their projects) and which projects have the best likelihood for a successful transition into library use.

### 1.3. Real-world characteristics

The personnel strain we experience every year through the discontinuity of students, 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)
- Solutions need to be integrated into existing USC Library operation. Independent ‘islands’ of solutions are not effective.

Most of the projects, so far have been considered a high success for our customer, the USC Library. Not only did they commit to pursue the projects for now three years in a row but also preparations for the next 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<br>Real-client project experience<br>Advanced software technology experience |
| Customers (librarians) | Useful applications<br>Advanced software technology understanding<br>Moderate time requirements |
| Faculty and staff | Educate future software engineering leaders<br>Better software engineering technology<br>Applied on real-client projects |

### 1.4. Outline

Experiences with this project have been summarized in detail in Ref. [8]. This paper builds strongly on Ref. [9] and as such further concentrates on how those projects helped us in integrating and validating software engineering technology.

The following section will summarize the software processes we used during the first year. Following that, we will discuss the improvements of the second and third year based on the lessons we learned from the previous ones. We will conclude this paper by discussing a few deeper aspects of software engineering technology integration which needs to supplement process integration in order to yield higher product maturity.

## 2. The first development projects

During the first year, we used a number of models to help develop the USC Library projects. The most important ones were the WinWin Spiral Model [7], the WinWin Negotiation Model [6], and COCOMO [3]. It is out of the scope of this paper to address them in detail. We will therefore only concentrate on the first one since it captures the development process the developers used, as well as the optimization process we used in validating their approaches.

### 2.1. Process models

The WinWin Spiral Model is a derivative of the original Spiral Model [4] which uses a cyclic approach to develop increasingly detailed elaboration of a software system (or process). 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. Ref. [21]), and successfully applied in numerous projects (e.g., Refs. [14,19]). However, some common difficulties have led to some further extensions to the 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 Fig. 2 [5]. First, identify the system or subsystem's key stakeholders. Second, identify the stakeholders' win conditions for the system or subsystem and, third, negotiate win–win reconciliation of the stakeholders' win conditions. What this means is that for each cycle we need to identify all important people (stakeholder) and their goals. We then need to reconcile their goals so that conflicts are resolved (the WinWin Negotiation Model addresses exactly this process).

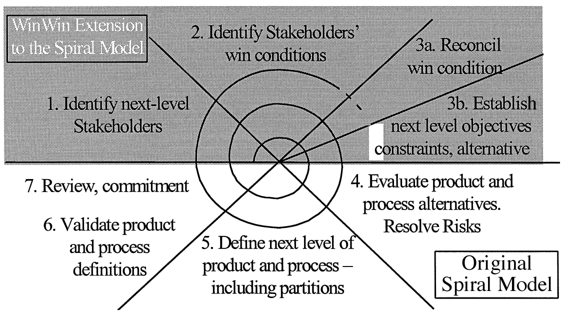

WinWin Extension to the Spiral Model
2. Identify Stakeholders' win conditions
3a. Reconcil win condition
1. Identify next-level Stakeholders
3b. Establish next level objectives constraints, alternative
7. Review, commitment
4. Evaluate product and process alternatives. Resolve Risks
6. Validate product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

Fig. 2. The WinWin spiral model.

We found that these steps indeed produced the key product and process objectives, constraints, and alternatives for the next version [6]. The overall stakeholder WinWin negotiation approach is similar to other team approaches 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 Ref. [7]. 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 milestone 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.

The initial milestone, the 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. This included a prototype, which was mostly done as part of the third cycle. Thus, the net result of the first semester's activity was to go from a 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 IOC 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

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

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

COTS and reused software, and appropriate operational readiness testing

- *Site preparation*, including facilities, equipment, supplies, and COTS vendor support arrangements

More usage information on the anchor points as well as their entry and exit criteria are described in Ref. [8].

## 2.3. 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, which was designed to keep track of changes during the negotiation. Besides the model implicit information, the tool also captured other usage activities in detail. For instance, Fig. 3 shows an example of the tool usage during the requirements negotiation period. The vertical axis shows the number of days the negotiation progressed

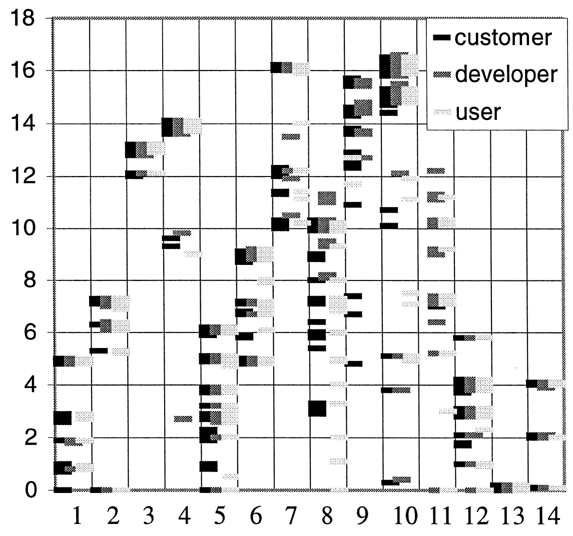


Fig. 3. Negotiation behavior—usage of WinWin over time.

and the horizontal axis shows the project teams. As it can be seen, stakeholders used the tool synchronously for the most time, however, some teams such as 8 and 10 also used it asynchronously.

- *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 LCO and LCA milestones, 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 in a quantitative manner, other than the grading of the LCO and LCA packages.
- *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 the form of questionnaires. So, 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. Basically, we asked them what they would do differently if they would have the chance to do it all over again.
- *Weekly Metrics*: Students were also asked to submit effort data on a weekly basis. The metrics, forms they had to fill out, described their daily 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 as follows.

### 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 marked the end of their involvement in the development process).

Having had a prototype which could be presented to the Library clients before the actual construction of the product was initiated was highly beneficial. However, in 1996 the librarians created the problem statement and participated in the WinWin requirements negotiation with the student teams before seeing the prototype. The prototypes yielded insightful surprises, but had 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 prototyping 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 being built in parallel with the WinWin requirements negotiation process and incorporated mostly user interface features. The second and third prototypes were build with the LCO and LCA milestones respectively. This stronger integration of prototyping into our development process resulted in the adaptation of Table 3 which got extended by the Prototyping milestone element (see Table 4).

## *3.2. Architecture review board (ARB)*

The end of the first semester (LCA milestone) in 1996 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 for all teams—thus enabling the teams to incorporate stakeholder feedback earlier on. We further structured those meetings more formally, 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. These were done in class and involved 6 of the 15 teams.

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. After all, their projects often needed to be integrated into the existing library systems and sometimes even with other Library projects.

Table 4
Contents of the prototype element for the LCO and LCA milestones

| Milestone element | Life cycle objectives (LCO) | Life cycle architecture (LCA) |
|---|---|---|
| Prototype | Exercise key usage scenarios<br>Resolve critical risks | Exercise range of usage scenarios<br>Resolve major outstanding risks |

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 [12]. Using UML, the teams were able to more strongly refine their software architectural description and using a design tool (like Rational Rose) 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 models could 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 we encountered during 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 was 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 which gave us extra some lead time for preparations.

The additional training sessions in the usage of WinWin, Rose, COCOMO, and other tools turned out to be highly effective. With that, the student teams had at least some tool experience before they used them in their 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 transition, operation, and maintenance. Most of the products, which got delivered after the first year, did not see operation in the USC Library.

During 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 were transitioned into library operations, and the other two have good prospects for transition after ongoing refinements are completed.

### 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; that of 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.

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. We also required the teams to post their documents on the class Web site a week before the ARB reviews. However, we 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 results can be seen in Table 5. We successfully reduced the document specification size by an average of 30–35%.

### 3.7. 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. Rational 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 was possible since many teams incorporated COTS products into their designs (e.g. the USC Library information system—SIRSI).
- Better structured student critiques and customer questionnaires because looking at first years questionnaires we found that there were some issues we would have liked to have feedback from all clients and students.

## 4. Improving the improvements

For the third year, we refined some of the issues we had discussed above. Particularly, we focused a lot of our attention onto the LCO/LCA documentation set again since we continued to experience critical consistency problems. Thus, we took the documentation guidelines and revised them very ex-

Table 5
Project characteristics

| Project characteristics | 1996–1997 | 1997–1998 | 1998–1999 |
|---|---|---|---|
| Architecture teams | 15 | 16 | 20 |
| Applications architected | 12 | 15 | 17 |
| Applications developed | 6 | 5 | 6 |
| Applications transitioned | 1 | 3–5 | N/A |
| LCO Spec pages | 160 | 110 | 114 |
| LCA Spec pages | 230 | 154 | N/A |
| Application types | Multimedia | Multimedia, text archives, ref. service, infrastructure | Multimedia, text archives, ref. service, infrastructure |

tensively in two ways. First, we elaborated in detail what is expected of each document all the way down to subsections. Second, we created a sample application which fitted into guidelines. For that sample applications, we modified one of our better projects of year two, the Hancock Museum Virtual Tour project.

The latter, although challenging, was not the real problem. The problem was to come up with a well integrated and detailed set of document guidelines. For one, if we would have just added needful things to our documentation guidelines, we would have run the risk of creating a document-centric development approach which would consume more development time than necessary in creating and maintaining documents—a luxury we did not have or need. On the other hand, if we would not have described our approach in enough detail, we could never have expected that all our development teams could follow it. We ended up describing why and what is needed for each section and in case it differed, describing the intended audiences, the creators (participants), high level dependencies with other parts of the documentation, and the tool support. We hope we found a reasonable compromise, but we will only know once this year's projects have been evaluated.

Another deficiency we would like to improve is our ability to make more accurate schedule estimations. As mentioned before, our teams use COCOMO (Constructive Cost Model) to plan and to track their projects. The accuracy of COCOMO can, however, be improved considerably if we could tailor it towards the library domain. In doing so we found that we did not gather all necessary data. This problem is being addressed this year and we hope what we can provide a calibrated version of COCOMO for next year. A major challenge with respect to that is also the issue of process maturity. Which CMM Level is 'our organization?' Obviously, we are doing many things which are required by various key process areas of the CMM, even process improvement (level 5) itself as this work shows. But does this mean that our development teams are CMM Level 5? This is probably not the case since we are not doing all required activities of the CMM. The reasons for that are two fold: (1) we cannot teach the student everything in software engineering plus doing an elaborate project in only two semesters time and (2) our projects team sizes are rather small with an average of five people per team and, thus, some activities of the CMM are not as important as others.

## 5. Integrating product and process

In optimizing the WinWin development process we are more and more confronted with the fact that just having a process is not enough. For one, we do not know how rigorous our students follow that process, and even if they do, our process (like many others) is not addressing another important aspect of software development. As Nuseibeh [17] wrote, ''the incremental development of software systems involves the detection and handling of inconsistencies.'' Obviously our process is able to handle changes (this is what the spiral model is know for in the first place) but nothing in the process model actually addressed how one can do this. When we talk about integrating process and product, we want to do this for two reasons:

1. To automatically find inconsistencies in the product (e.g. does not conform to requirements) and in the process (e.g. actual process did not follow planned process).
2. To guide the developer, which implies that the process is able to adapt itself based on the state of the product. So, for instance, if an inconsistency in the product is encountered, the process could suggest options on how to resolve it.

To address this problem, we are also extending our models. The most significant result, so far, has been the consolidation of many of our models into an integrated conceptual model called MBASE (Model-Based Architecting and Software Engineering) [11].

The MBASE model concentrates on avoidance of model clashes between process model, success model, product model, and property model. Fig. 4 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 the Stakeholders' win conditions as input and delivers the WinWin Agreements as an output. The Agreements in turn may be used to create a variety of documents, such as the Requirements Document and so forth.

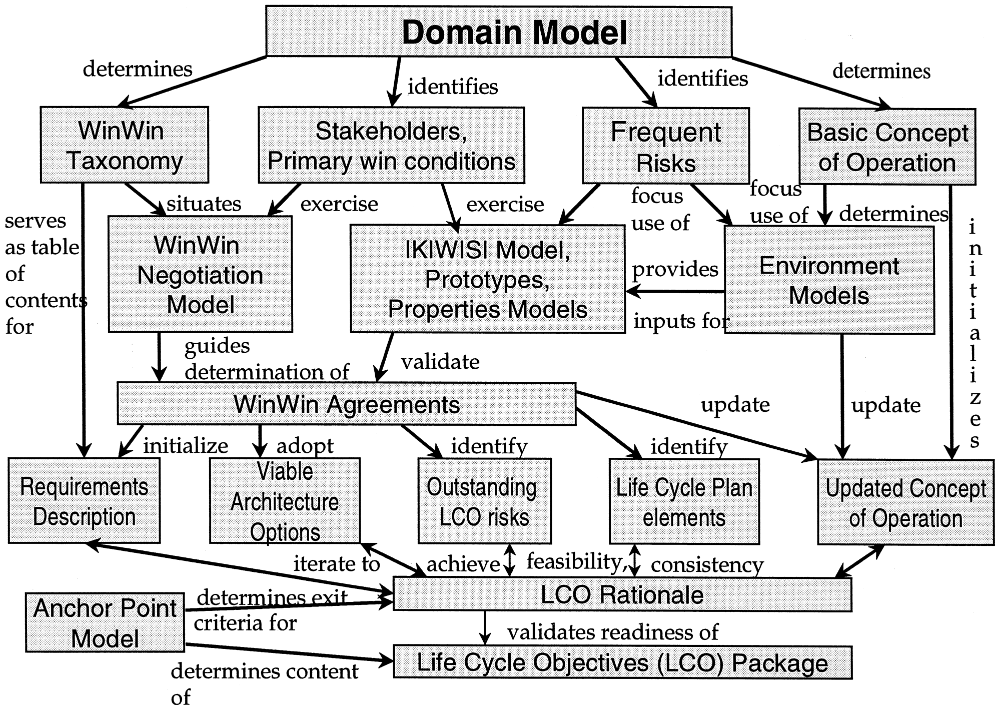


Fig. 4. Detailed process for LCO stage.

To make MBASE work and to truly create an integrated model, we have to do more than just providing a process model; we need to be somehow able to integrate development models in general. Take, for instance, the documentation issue we discussed above. We mentioned that we are encountering problems when it comes to the conceptual integrity of the documents. This is partially because they are developed concurrently by a number of people and it is partially because sometimes they are not clear what they mean and what they are for. The latter could be a problem related to the former since the wrong person could be working on the document.

Yet another problem is the redundancy between those documents. We had mentioned that it is our goal to minimize the redundancy but it is impossible to eliminate it. The reason for that is simple—each document needs to be independent enough so that it makes sense on its own and can be read independently without having to jump between documents. On the other hand, we could create those documents using hypertext and the World-Wide-Web (WWW) and this way we would avoid duplicating the same information for the most part. However, the result may be hundreds if not thousands of little clusters of information, which are somehow glued together but impossible to read in sequence.

A possible way of solving this problem is emphasizing model-based development (as in MBASE). We are moving towards using the MBASE document model to store those little clusters of information and on top of it to define views, e.g., the Requirements document view, which use pieces of this model. This way the reader would get the impression that he or she is reading a standalone document. Changes, which could be done in the model itself or in any of its views, could then be automatically propagated across all other views. This form of model-based development may look as depicted in Fig. 5 (see also Ref. [13]).

As can be seen in Fig. 5, views are nothing more than abstractions of relevant information from the system model and they present those bits of information in some meaningful way to the user (developer, customer, etc.). Thus, when we talk about the need for tightly coupled views we are really talking about

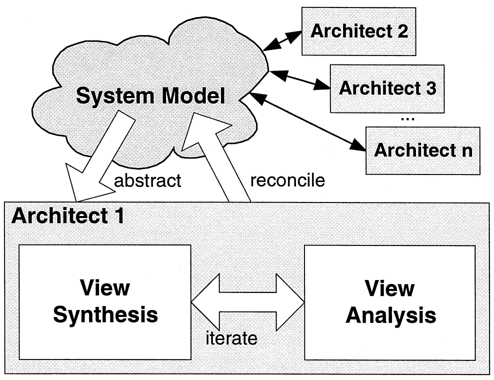


Fig. 5. Model-based development.

the need of having an integrated model which is adequate in representing those views. The stakeholders (e.g. developer or customer) can then derive views (e.g. requirements document) from that model, fill in the missing blanks, and reconcile the changes with the model. This means that all information about a software system is captured with as little redundancy as possible in the model, even though, the views, which are derived from that model, may repeatedly use the same information and, thus, may exhibit redundancy. In that respect we are talking of the model as being a *View Independent Representation*. We are exploring this form of the view integration problem with particular emphasis on software architectures and UML in Ref. [13].

## 6. Conclusions

Our collaboration with the Library continues to be very beneficial for us all in developing and evaluating new ideas. We are currently involved in observing this year's students so that we may yet improve the process a bit further for next year. To some degree, we need to adapt and refine our existing process but we also need to integrate our models more tightly as described above. Next to that, we also see the following areas as equally rewarding topics for us to explore further.

- Combining the Best of Standards: Our original standards were closely tailored to DoD standards (e.g., DoD 498). We are now involved in integrating new standards such as the IEEE/EIA 12207.1-1997 [15] into our process model.
- Product Lines: Since most of our projects are in the library domain we would like to gain better insights on how to reuse components or even on how to reuse entire application frameworks. This would enable us to use some teams in the future to provide some form of product line support for other teams.
- Integrating the Rational Unified Process [16] and the Unified Software Management [20] into our model/process.

The library projects were and will continue to be a testing field for these kinds 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—and other cultures. We also see this testing field as an effective way of convincing our affiliates and the software industry in general of the usefulness of our models since we are able to show their effectiveness in a very realistic project environment.

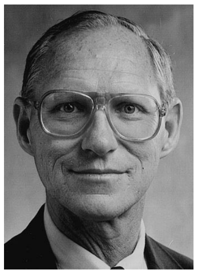
## Acknowledgements

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: Bellcore, Boeing, Electronic Data Systems, FAA, GDE Systems, Hughes Aircraft, Institute for Defense Analysis, Litton Systems, Lockheed Martin, Loral Federal Systems, Lucent, MCC, Motorola, Network Programs, Northrop Grumman, Rational Software, Raytheon, 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.

## References

[1] AT&T, Best Current Practices: Software Architecture Validation, AT&T, Murray Hill, NJ, 1993.
[2] V.R. Basili, R.W. Selby, D.H. Hutchens, Experimentation in Software Engineering, IEEE Transactions on Software Engineering, July 1986, pp. 733–743.

[3] B.W. Boehm, Software Engineering Economics, Prentice-Hall, 1981.
[4] B.W. Boehm, A spiral model of software development and enhancement, Computer 21 (5) (1988) 61–72.
[5] B.W. Boehm, P. Bose, 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 1994.
[6] B.W. Boehm, P. Bose, E. Horowitz, M.J. Lee, Software Requirements as Negotiated Win Conditions, Proceedings of ICRE, 1994, pp. 74–83.
[7] B.W. Boehm, Anchoring the software process, IEEE Software 13 (4) (1996) 73–82.
[8] B.W. Boehm, A.F. Egyed, J. Kwan, R. Madachy, D. Port, A. Shah, Using the WinWin Spiral Model: A Case Study, IEEE Computer, July 1998.
[9] B.W. Boehm, A.F. Egyed, Improving the Life-Cycle Process in Software Engineering Education, Proceedings of the First European Software Day, 24th Euromicro Conference, August 1998.
[10] B.W. Boehm, A.F. Egyed, D. Port, A. Shah, J. Kwan, R. Madachy, A Stakeholder Win–Win Approach to Software Engineering Education, Annals of Software Engineering, 1999, to appear.
[11] B.W. Boehm, D. Port, Conceptual Modeling Challenges for Model-Based Architecting and Software Engineering (MBASE), Proceedings, Symposium on Conceptual Modeling, Springer-Verlag, 1998.
[12] G. Booch, I. Jacobson, J. Rumbaugh, The Unified Modeling Language for Object-Oriented Development, Documentation set, version 1.0, Rational Software, 1997.
[13] A.F. Egyed, Integrating Architectural Views in UML, Qualifying Report, University of Southern California, Center for Software Engineering, Los Angeles, CA, 1999 (to appear).
[14] T.P. Frazier, J.W. Bailey, The costs and benefits of domain-oriented software reuse: Evidence from the STARS demonstration projects, IDA Paper P-3191, Institute for Defense Analysis, 1996.
[15] IEEE, Industry Implementation of International Standard ISO/EIC 12207: 1995, IEEE/EIA 12207.1-1997, April 1998.
[16] P.B. Kruchten, The Rational Unified Process, Addison-Wesley, 1999.
[17] B. Nuseibeh, Computer-Aided Inconsistency Management in Software Development, Technical Report DoC 95/4, Department of Computing, Imperial College, London SW7 2BZ, 1995.
[18] M.C. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis, The Capability Maturity Model—Guidelines for Improving the Software Process, Addison-Wesley, 1995.
[19] W.E. Royce, TRW's Ada Process Model for Incremental Development of Large Software Systems, Proceedings, ICSE 12, IEEE/ACM, March 1990, pp. 2–11.
[20] W.E. Royce, Unified Software Management, Addison-Wesley, Reading, MA, 1998, to be published.
[21] Software Productivity Consortium, Process Engineering with the Evolutionary Spiral Process Model, SPC-93098-CMC, version 01.00.06, Herndon, VA, 1994.



Barry Boehm is the TRW Professor of Software Engineering and Director of the Center for Software Engineering at the University of Southern California. His current research involves the WinWin groupware system for software requirements negotiation, architecture-based models of software quality attributes, and the COCOMO II cost-estimation model. 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 an AIAA Fellow, an ACM Fellow, an IEEE Fellow, and a member of the National Academy of Engineering.

Alexander Egyed is a PhD student at the Center for Software Engineering at the University of Southern California. His research interests are in software architecture, design, and requirements elicitation. He received a Diplom-Ingenieur in Informatics from the Johannes Kepler University in Linz, Austria and a MS in Computer Science from the University of Southern California. He is a student member of the IEEE and the ACM.