Using the WinWin Spiral Model: A Case Study

Fifteen teams used the WinWin spiral model to prototype, plan, specify, and build multimedia applications for USC's Integrated Library System. The authors report lessons learned from this case study and how they extended the model's utility and cost-effectiveness in a second round of projects.

Barry Boehm
Alexander Egyed
Julie Kwan
Dan Port
Archita Shah
University of Southern California

Ray Madachy
Litton Data Systems and University of Southern California

At the 1996 and 1997 International Conferences on Software Engineering, three of the six keynote addresses identified negotiation techniques as the most critical success factor in improving the outcome of software projects. At the USC Center for Software Engineering, we have been developing a negotiation-based approach to software system requirements engineering, architecture, development, and management. Our approach has three primary elements:

- *Theory W*, a management theory and approach, which says that making winners of the system's key stakeholders is a necessary and sufficient condition for project success.[1]
- *The WinWin spiral model*, which extends the spiral software development model by adding Theory W activities to the front of each cycle. The sidebar "Elements of the WinWin Spiral Model" describes these extensions and their goals in more detail.
- *WinWin*, a groupware tool that makes it easier for distributed stakeholders to negotiate mutually satisfactory (win-win) system specifications.[2]

In this article, we describe an experimental validation of this approach, focusing on the application of the WinWin spiral model. The case study involved extending USC's Integrated Library System to access multimedia archives, including films, maps, and videos. The Integrated Library System is a Unix-based, text-oriented, client-server COTS system designed to manage the acquisition, cataloging, public access, and circulation of library material. The study's specific goal was to evaluate the feasibility of using the WinWin spiral model to build applications written by USC graduate student teams. The students developed the applications in concert with USC library clients, who had identified many USC multimedia archives that seemed worthy of transformation into digitized, user-interactive archive management services.

The study showed that the WinWin spiral model is a good match for multimedia applications and is likely to be useful for other applications with similar characteristics—rapidly moving technology, many candidate approaches, little user or developer experience with similar systems, and the need for rapid completion. The study results show that the model has three main strengths.

- *Flexibility*. The model let the teams adapt to accompanying risks and uncertainties, such as a rapid project schedule and changing team composition.
- *Discipline*. The modeling framework was sufficiently formal to maintain focus on achieving three main, or "anchor-point," milestones: the life-cycle objectives, the life-cycle architecture, and the initial operational capability. (Table A in the sidebar describes these milestones.)
- *Trust enhancement*. The model provided a means for growing trust among the project stakeholders, enabling them to evolve from adversarial, contract-oriented system development approaches

0018-9162/98/$10.00 © 1998 IEEE

Elements of the WinWin Spiral Model

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

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

Since its creation, the spiral model has been extensively elaborated[2] and successfully applied in numerous projects.[3,4] However, some common difficulties led USC-CSE and its affiliate organizations to extend the model to the WinWin spiral model described in the main text.

Negotiation front end

One difficulty was determining where the elaborated objectives, constraints, and alternatives come from. The WinWin spiral model resolves this by adding three activities to the front of each spiral cycle, as Figure A shows.[5]

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

We have found in experiments with a bootstrap version of the WinWin groupware tool that these steps do indeed produce the key product and process objectives, constraints, and alternatives for the next version.[6] The model includes a stakeholder WinWin negotiation approach that is similar to other team approaches for software and system definition such as gIBIS, Viewpoints, Participatory Design, and Joint Application Design. However, unlike these and other approaches, we use the stakeholder win-win relationship as the success criterion and organizing principle for software and system definition. Our negotiation guidelines are based on the Harvard Negotiation Project's techniques.[7]

Process anchor points

Another difficulty in applying the spiral model across an organization's various projects was that the organization has no common reference points for organizing its management procedures, cost and schedule estimates, and so on. This is because the cycles are risk driven, and each project has different risks. In attempting to work out this difficulty with USC-CSE's industry and government affiliates using our Cocomo II cost model, we found a set of three process milestones, or *anchor points*,[8] which we could relate to both the completion of spiral cycles and to the organization's major decision milestones.

The *life-cycle objectives* (LCO) and the *life-cycle architecture* (LCA) milestones ratify the stakeholders' commitment to a feasible and consistent package of the six key milestone elements shown in Table A for the LCO anchor point.

The LCO version focuses on establishing a sound business case for the package. It need only show that there is at least one feasible architecture.

The LCA version commits to a single choice of architecture and elaborates it to the point of covering all major sources of risk in the system's life cycle.[8] The LCA is the most

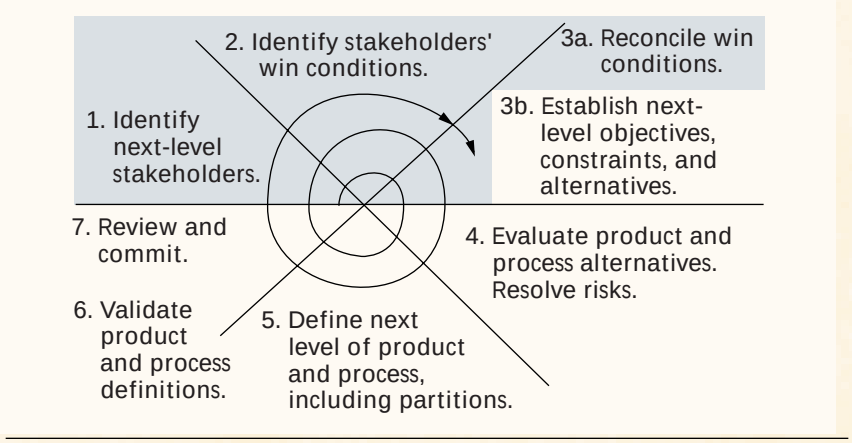


Figure A. How the WinWin spiral model differs from the original spiral model. The new model adds front-end activities (blue) that show where objectives, constraints, and alternatives come from. This lets users more clearly identify the rationale involved in negotiating win conditions for the product.

toward methods that were mutually supportive and cooperative.

From lessons learned during the case study, we identified several possible enhancements, some of which we made. We then used the enhanced model on 16 projects in the following year. The second-year projects overcame many of the weaknesses in the first-year projects. We are incorporating improvements identified by student critiques and are planning third-year projects. Industry is also looking at the WinWin spiral model. Companies such as Rational Inc. have already adopted several elements of the WinWin spiral model as part of their project management and product life-cycle processes.

MODEL APPLICATION

We applied the WinWin spiral model in four cycles:

- *Cycle 0*. Determine the feasibility of an appropriate family of multimedia applications.
- *Cycle 1*. Develop life-cycle objectives (LCO milestone), prototypes, plans, and specifications for individual applications and verify the existence of at least one feasible architecture for each application.
- *Cycle 2*. Establish a specific, detailed life-cycle architecture (LCA milestone), verify its feasibility, and determine that there are no major risks in satisfying the plans and specifications.
- *Cycle 3*. Achieve a workable initial operational capability (IOC milestone) for each project

critical milestone in the software system's life cycle. As an analogy, it is similar to the commitment you make in getting married (just as LCO is like getting engaged and IOC like having your first child).

The *initial operational capability*, or IOC, anchor point has three key elements:[8]

- 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.
- User, operator, and maintainer preparation, including selection, team building, training, and other qualifications for familiarization use, operations, or maintenance.

We found that the LCO and LCA milestones are highly compatible with the successful architecture review board practice pioneered by AT&T and Lucent Technologies.[9] We used board sessions about 20 percent of the time in the first-year projects and 100 percent of the time in the second-year projects, with much better results.

References

1. B. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, May 1988, pp. 61-72.
2. "Process Engineering with the Evolutionary Spiral Process Model: Version 01.00.06," Tech. Report SPC-93098-CMC, Software Productivity Consortium, Herndon, Va., 1994.
3. W. E. Royce, "TRW's Ada Process Model for Incremental Development of Large Software Systems," *Proc. 12th Int'l Conf. Software Eng.*, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 2-11.
4. T. Frazier and J. Bailey, "The Costs and Benefits of Domain-Oriented Software Reuse: Evidence from the STARS Demonstration Projects," IDA Paper P-3191, Institute for Defense Analyses, Alexandria, Va., 1996.
5. B. Boehm and P. Bose, "A Collaborative Spiral Software Process Model Based on Theory W," *Proc. Int'l Conf. Software Process*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 59-68.
6. B. Boehm et al., "Software Requirements as Negotiated Win Conditions," *Proc. Int'l Conf. Requirements Eng.*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 74-83.
7. R. Fisher and W. Ury, *Getting to Yes*, Penguin Books, New York, 1981.
8. B. Boehm, "Anchoring the Software Process," *IEEE Software*, July 1996, pp. 73-82.
9. "Best Current Practices: Software Architecture Validation," AT&T, Murray Hill, N.J. 1993.

Table A. Contents of the LCO milestone.

Milestone element					
Definition of operational concept	**Definition of system requirements**	**Definition of system and software architecture**	**Definition of life-cycle plan**	**Feasibility rationale**	**System prototype(s)**
Top-level system objectives and scope Environment parameters and assumptions Evolution parameters Operational concept Operations and maintenance scenarios and parameters Organizational life-cycle responsibilities (stakeholders)	Top-level functions, interfaces, quality attribute levels, including: Growth vectors Priorities Stakeholders' concurrence on essentials	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	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 (Why, What, When, Who, Where, How, How Much?) by stage	Assurance of consistency among elements above Via analysis, measurement, prototyping, simulation, etc. Business case analysis for requirements, feasible architectures	Exercise key usage scenarios Resolve critical risks

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

We used Theory W in all the cycles, but we used the WinWin groupware tool in Cycle 1 only, because this is where it currently works best.

Cycle 0: Application family

From 1993 to 1996, the USC Center for Software Engineering (CSE) experimented with teaching the WinWin spiral model in its master's software engineering course, taught by Barry Boehm. The experiments involved using hypothetical applications, one of which was an advanced library application. Some of the library staff became interested in having the CSE students develop useful USC library applications.

The CSE in turn had been looking for a source of new applications on which to test the WinWin spiral model. So in the summer of 1996 we met with some of the library staff to explore respective win conditions and to determine if we could identify a feasible set of life-cycle objectives for a family of USC library applications. Table 1 summarizes the win conditions for the three primary stakeholders: the library information technology community; the library operations community (including users); and the CSE.

As the table indicates, the library information technology community was energized by their dean's vision to accelerate the libraries' transition to digital

Table 1. Win conditions for the three primary stakeholders in the case study.

Library Information Technology Community	Library Operations Community	Center for Software Engineering
Accelerated transition to digital library capabilities; vision of Dean of the University Libraries	Continuity of service	Similarity of projects (for fairness, project management)
Evaluation of emerging multimedia archiving and access tools	No disruption of ongoing transition to SIRSI-based Library Information System	Reasonable match to the WinWin spiral model
Empowering library multimedia users	Career growth opportunities for system administrators	15-20 projects at 5-6 students per team
Enhancement of library staff capabilities in digital library services	No disruption of USC network operations and services	Achieve a meaningful life-cycle architecture in one semester
Leveraging of limited budget for advanced applications	More efficient operations via technology	Achieve a meaningful initial operational capability in two semesters
		Adequate network, computer, and infrastructure resources

capabilities. However, there was little budget for evaluating emerging multimedia technology and developing exploratory applications.

The library operations community and its users were already undergoing a complex transition to the new Integrated Library System. They were continually looking for new technology to enhance their operations. But they were also highly sensitive to the risks of disrupting services, and they had limited resources to experiment in new areas.

The biggest risk identified in Cycle 0 was the risk of having too many different applications and losing control of the project. Achieving a meaningful IOC in two semesters, a win condition for CSE, meant following a rapid project schedule. Because the students would be unfamiliar with both one another and with their library applications and clients, they could easily go off in all directions. We resolved this risk by focusing on a single application area—library multimedia archive services—and by developing a common domain model and set of product guidelines for all teams to follow.

Cycle 1: Application life-cycle objectives

Figure 1 shows the project guidelines we negotiated with the library staff during Cycle 0 and provided to the CS students on the first day of class. The guidelines allowed 2.5 weeks for the students to organize themselves into teams and 11.5 weeks to complete the life-cycle objective and life-cycle architecture milestones.

We also gave each project some guidelines for developing five documents (in the artifacts list under "Project Objectives" in Figure 1), including recommended page budgets. Each team had to develop two versions, one for the LCO milestone and an elaboration for the LCA milestone. To ensure that everyone used a common development process, we gave the teams a sample multimedia archive prototype and a domain model for a typical information archive extension. The domain model, in Figure 2, identifies the key stakeholders involved in such systems and key concepts like the system boundary, the boundary between the system being developed and its environment.

The project guidelines and domain model were key to the teams' rapid progress because they provided a common development perspective. The course lectures followed the WinWin spiral model. We began with overviews of the project artifacts (in Figure 1 under "Project objectives") and how they fit together. We continued with a discussion of the key planning and organizing guidelines. In later lectures, we provided more detail on the project artifacts and had guest lectures on library operations and the SIRSI system and on technological aspects such as user interface design and multimedia system architecture.

We focused each team during Cycle 1 by having them use the WinWin groupware tool for requirements negotiation.[2] "WinWin user negotiations" in Figure 1 identifies the four key forms in the WinWin negotiation model (win conditions, issues, options, and agreements), and their relationships. It also summarizes the stakeholder roles (developer, customer, and user) to be played by the team members. To minimize disruption to library operations, we had the operational concept and requirements team members enter the user artifacts, rather than the librarians themselves.

Figure 3a shows the final list of applications and the teams required to develop them. We ended up with 12 applications and 15 development teams, comprising both on- and off-campus students. We let the project teams select their own members to mitigate the risk of forming teams with incompatible people and philosophies. Most teams had six people.

Figure 3b shows two problem statements prepared by the library clients. These statements are much less detailed than a typical requirements set in an industrial application. The team had to go from short statements like this to a consistent set of prototypes, plans, and specifications (typically 200 pages) in 11 weeks.

To help them organize and navigate the WinWin artifacts and control the associated terminology, we gave each team a domain taxonomy and guidelines for relating the taxonomy elements to elements of the requirements specification. Figure 4 shows part of the taxonomy and guidelines.

Figure 5 shows the look and feel of the WinWin tool. In the lower right is a win condition form entered

Project objectives

Create the artifacts necessary to establish a successful life-cycle architecture and plan for adding a multimedia access capability to the USC Library Information System. These artifacts are

1. An operational concept definition
2. A system requirements definition
3. A system and software architecture definition
4. A prototype of key system features
5. A life-cycle plan
6. A feasibility rationale, assuring the consistency and feasibility of items 1-5.

Team structure

Each of the six team members will be responsible for developing the LCO and LCA versions of one of the six project artifacts. In addition, the team member responsible for the feasibility rationale will serve as project manager with the following primary responsibilities:

- Ensure consistency among the team members' artifacts (and document this in the rationale).
- Lead the team's development of plans for achieving the project results and ensure that project performance tracks the plans.

Project approach

Each team will develop the project artifacts concurrently, using the WinWin spiral approach defined in the article "Anchoring the Software Process." There will be two critical project milestones: the life-cycle objectives (LCO) and life-cycle architecture (LCA). The LCA package should be sufficiently complete to support development of an initial operational capability (IOC) version of the planned multimedia access capability by a CS577b student team during the spring 1997 semester. The life-cycle plan should establish the appropriate size and structure of the development team.

WinWin user negotiations

Each team will work with a representative of a community of potential users of the multimedia capability (art, cinema, engineering, business, etc.) to determine that community's most significant multimedia access needs and to reconcile these needs with a feasible implementation architecture and plan. The teams will accomplish this reconciliation by using the USC WinWin groupware support system for requirements negotiation. This system provides WinWin forms for stakeholders to express their win conditions for the system, to define issues dealing with conflicts among win conditions, to support options for resolving the issues, and to consummate agreements to adopt mutually satisfactory (win-win) options.

There will be three stakeholder roles:

- *Developer.* The architecture and prototype team members will represent developer concerns, such as the use of familiar packages, stability of requirements, availability of support tools, and technically challenging approaches.
- *Customer.* The plan and rationale team members will represent customer concerns, such as the need to develop an IOC in one semester, limited budgets for support tools, and low-risk technical approaches.
- *User.* The operational concept and requirements team members will work with their designated user-community representative to represent user concerns, such as particular multimedia access features, fast response time, friendly user interface, high reliability, and flexibility of requirements.

Major milestones

September 16	All teams formed
October 14	WinWin negotiation results
October 21, 23	LCO reviews
October 28	LCO package due
November 4	Feedback on LCO package
December 6	LCA package due, individual critique due

Individual project critique

The project critique is to be done by each individual student. It should be about 3-5 pages, and should answer the question, "If we were to do the project over again, how would we do it better—and how does that relate to the software engineering principles in the course?"

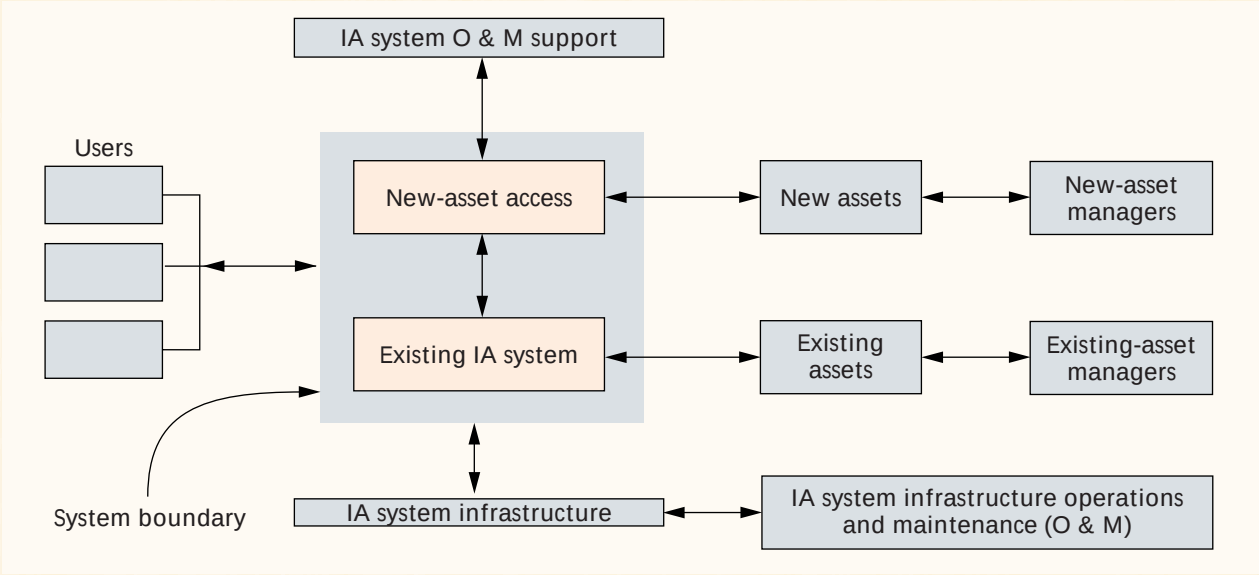
Figure 1. Guidelines given to the 15 teams on how to conduct their respective multimedia archive projects. Because each project team received the same guidelines, the teams were able to progress rapidly in specifying and building the applications.

by one of the team members on the Hancock Library photo archive project, expressing the need to accommodate future upgrades, such as different image formats. The graph at the top shows how the win condition "swong-WINC-5" is linked to other WinWin forms such as issues, options, and agreements. The taxonomy helps categorize these forms into common concerns, such as those that affect user controls.

The WinWin negotiation period took longer than we expected for several reasons. We underestimated the amount of WinWin training needed and the complexities of supporting 15 simultaneous negotiations, some with mixes of on- and off-campus negotiators. As a result, we moved the deadline for completing the WinWin negotiations and the LCO packages back a week. Fortunately, the LCO packages were good enough to let us make up that time in the next cycle.

Under the revised schedule, all 15 teams delivered their LCO packages on time. The degree of completeness was generally appropriate, but components

System block diagram
This diagram shows the usual block diagram for extensions providing access to new information archive assets from an existing information archive (IA) system:



The system boundary focuses on the automated applications portion of the operation and defines such external entities as users, operators, maintainers, assets, and infrastructure (campus networks, etc.) as part of the system environment. The diagram abstracts out such additional needed capabilities as asset catalogs and direct user access to O&M support and asset managers. Some stakeholder roles and responsibilities include:

- *Asset managers.* Furnish and update asset content and catalog descriptors. Ensure access to assets. Provide accessibility status information. Ensure asset-base recoverability. Support problem analysis, explanation, training, instrumentation, operations analysis.
- *Operators.* Maintain high level of system performance and availability. Accommodate asset and services growth and change. Protect stakeholder privacy and intellectual property rights. Support problem analysis, explanation, training, instrumentation, operations analysis.
- *Users*. Obtain training. Access system. Query and browse assets. Import and operate on assets. Establish, populate, update, and access asset-related user files. Comply with system policies. Provide feedback on use.
- *Application software maintainer*. Perform corrective, adaptive, and perfective (tuning, restructuring) maintenance on software. Analyze and support prioritization of proposed changes. Plan, design, develop, and verify selected changes. Support problem analysis, explanation, training, instrumentation, and operations analysis.
- *Service providers (network, database, or facilities management services)*. Roles and responsibilities similar to asset managers.

Figure 2. Domain model for extending an information archive system. The domain model and the guidelines in Figure 1 helped give the 15 teams a unified perspective of project development.

often had serious inconsistencies in assumptions, relationships, and terminology. Most teams had planned time for members to review each others' artifacts, but most individual members ended up using that time to finish their artifacts. Some concepts—such as the nature of the system boundary, organizational relationships, and the primary goal of the life-cycle plan—caused problems for students without industrial experience. We covered these concepts in more depth in subsequent course lectures.

Cycle 2: Application life-cycle architectures

In Cycle 2, the teams chose a specific life-cycle architecture for their applications and elaborated the content of their LCO artifacts to the level of detail required for the LCA milestone. This included responding to the instructors' comments on their LCO packages. The most frequent problems were inconsistencies among the artifacts, failure to specify quality attributes, a general misunderstanding about the application's scope (the system boundary in Figure 2a) and the inability to recognize that the plan was to focus on the development activities in Cycle 3.

Because of delays and changes in prototyping equipment, the teams developed their prototypes in Cycle 2. This had the unfortunate effect of destabilizing some of the WinWin agreements and product requirements. Once the library clients saw the prototypes, they wanted to change the requirements (the IKIWISI—I'll know it when I see it—syndrome). In the following year, we had the teams do the initial prototyping and WinWin negotiations concurrently.

On the positive side, the prototypes generally expanded the librarians' perceptions of what the teams could produce. The librarian who proposed the Edgar corporate data problem was amazed with the end product, which built on the seemingly simple text-formatting problem and delivered a one-stop Java site that synthesized several kinds of business information. She commented in her evaluation memo:

Team	Application
1.	Stereoscopic slides
2.	Latin American pamphlets
3,5.	Edgar corporate data
4.	Medieval manuscripts
6,10.	Hancock Library photo archive
7.	Interactive TV courseware delivery
8,11.	Technical reports archives
9.	Student film archive
12.	Student access to digital maps
13.	Los Angeles regional history photos
14.	Korean-American museum
15.	Urban planning documents

(a)

Medieval manuscripts
I am interested in how to scan medieval manuscripts so that a researcher could both read the content and study the scribe's hand, special markings, and so on. A related issue is how to transmit such images.

Edgar corporate data
Increasingly the government is using the WWW as a tool for dissemination of information. Two much-used sites are the Edgar database of corporate information (http://www.sec.gov/edgarhp.htm) and the Bureau of the Census (http://www.census.gov). Part of the problem is that some information (particularly that at the Edgar site) is available only as ASCII files. For textual information, the formatting of statistical tables is often lost in downloading, e-mailing, or transferring to statistical programs. While this information is useful for the typical library researcher, it is often too much trouble to put it in a usable format.

(b)

Figure 3. (a) Proposed multimedia applications and (b) two problem statements prepared by the library clients. The numbers in (a) designate the teams that designed the application. Because we had 15 teams and 12 applications, some library clients agreed to work with two teams. From statements like those in (b), the teams had to generate detailed specifications in 11 weeks.

1. Operational Modes
    - 1.1 Classes of Service (research, education, general public)
    - 1.2 Training
    - 1.3 Graceful Degradation and Recovery
2. Capabilities
    - 2.1 Media Handled
        - 2.1.1 Static (text, images, graphics, etc.)
        - 2.1.2 Dynamic (audio, video, animation, etc.)
    - 2.2 Media Operations
        - 2.2.1 Query, Browse
        - 2.2.2 Access
        - 2.2.3 Text Operations (find, reformat, etc.)
        - 2.2.4 Image Operations (zoom in/out, translate/rotate, etc.)
        - 2.2.5 Audio Operations (volume, balance, forward/reverse, etc.)
        - 2.2.6 Video/Animation Operations (speedup/slowdown, forward/reverse, etc.)
        - 2.2.7 Adaptation (cut, copy, paste, superimpose, etc.)
        - 2.2.8 File Operations (save, recall, print, record, etc.)
        - 2.2.9 User Controls
    - 2.3 Help
    - 2.4 Administration
        - 2.4.1 User Account Management
        - 2.4.2 Use Monitoring and Analysis
3. Interfaces
    - 3.1 Infrastructure (SIRSI, UCS, etc.)
    - 3.2 Media Providers
    - 3.3 Operators
4. Quality Attributes

    ⋮

The taxonomy serves as a requirements checklist and navigation aid:
The taxonomy elements map onto the requirements description table of contents in the course notes.
Every WinWin artifact should point to at least one taxonomy element (modify elements if appropriate).
Every taxonomy element should be considered as a source of potential stakeholder win conditions and agreements.

Figure 4. Part of the domain taxonomy and use guidelines given to each project team. The taxonomy specializes the WinWin tool to the stakeholders' domain, and serves as a checklist for completing the negotiation process. It also helped the teams organize the WinWin forms and relate them to the requirements specification.

> "[The team] obviously looked beyond the parameters of the problem and researched the type of information need the set of data meets. My interactions with the team were minimal, not because of any difficulty, but because as a group they had a synergy and grasped the concepts presented to them. The solution the team came up with was innovative, with the potential to be applied to other, similar problems."

Other library clients were also very satisfied with the value added relative to their time invested.

The teams were able to surmount several challenges characteristic of real-world projects. For example, they could not use the Integrated Library System's test server for their prototypes because it was needed in transitioning from the old system to the new Integrated Library System. There were also delays in arranging for a suitable alternative Web server. At times librarians could not provide input on critical decisions, which led to extra rework. Inevitable per-

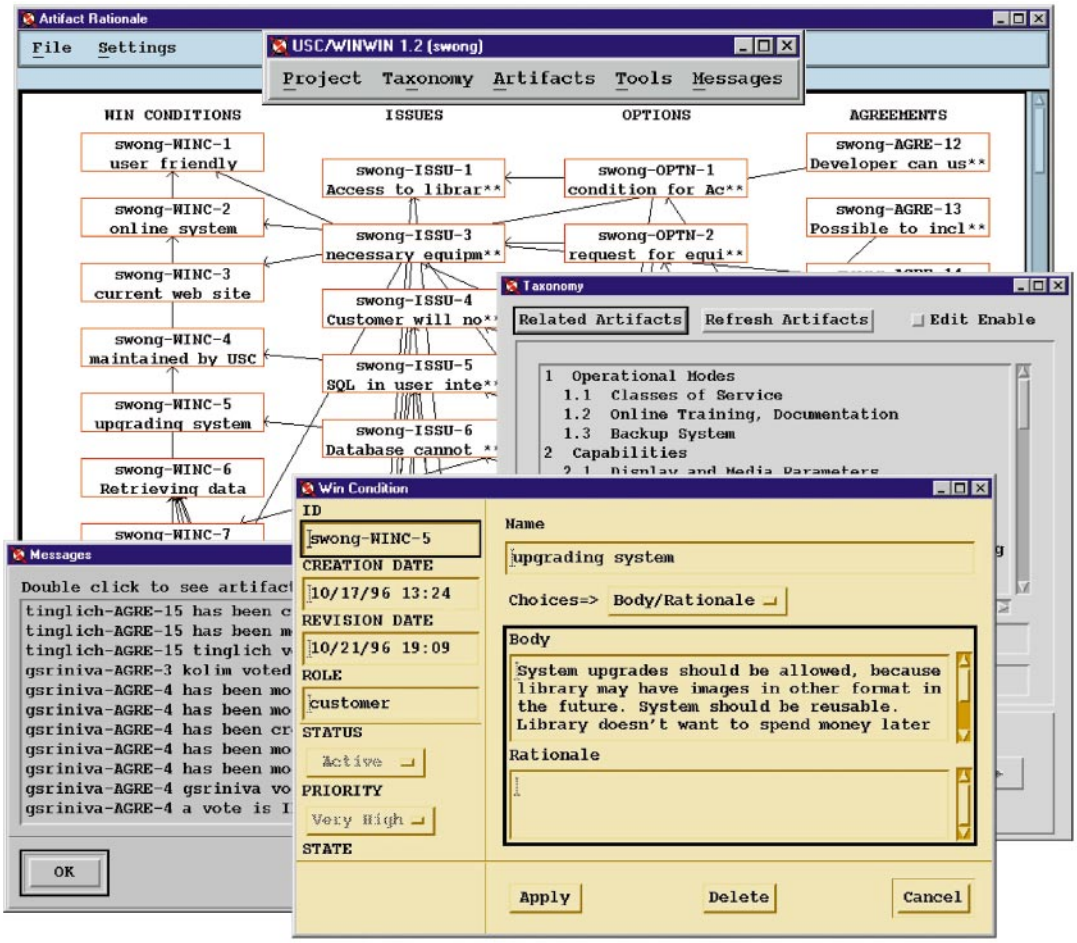


Figure 5. Sample screens from the WinWin groupware tool. The artifact rationale window (upper left) lets users immediately see links among the project stakeholders' win conditions, issues, options, and negotiated agreements. The screen in the lower right expands one of these forms, a stakeholder's win condition. The current negotiation outline is the taxonomy window (middle right) and the latest changes are listed in the message window (bottom left).

sonnel conflicts arose within the 15 teams. However, we were able to minimize conflicts within a team, in large part because the teams were self-selected.

The WinWin spiral model's mix of flexibility and discipline let the project teams adapt to these challenges while staying on schedule. In particular, the use of risk management and a continuously evolving top *n* risk list[3] helped the teams focus their effort on the most critical success factors for their projects.

Another difficulty was maintaining consistency across multiple product views. The guidelines we gave the students were from the course textbook,[4] evolving commercial standards like J-STD-016-1995, and object-oriented methods, particularly the Booch method and Object Modeling Technology. The views included system block diagrams, requirements templates, use scenarios, physical architecture diagrams, class hierarchies, object interaction diagrams, dataflow diagrams, state-transition diagrams, data descriptions, and requirements traceability relations. Each had its value, but the overall set was both an overkill and weakly supported by integrated tools. In the following year, we used a more concise and integrated set of views based on the Rational Unified Modeling Language and tool set.[5]

Cycle 3: Initial operational capability

A major challenge for Cycle 3 was that many students involved in Cycles 1 and 2 during the fall 1996 Software Engineering I course, which was a core course for an MS in computer science, did not take Software Engineering II in spring 1997 because it was not a core course. Thus, we were able to continue only six projects during Cycle 3, involving 28 students and eight applications. The projects that continued were driven by the project experience of the students who reenrolled, rather than by the priorities of the librarians.

Only one team retained most of its LCO/LCA participants for Cycle 3. The other teams had to work with a mix of participants with varying project backgrounds. This was particularly challenging when we had to integrate teams that had produced different LCA artifacts for the same application. In two cases, the instructors had to persuade students to join different teams because they continued to fight about whose architecture was better. Other conflicts developed within teams in which some members had extensive LCA experience on the application and others had none. In one case, experienced members exploited those less experienced; in another case, the reverse happened.

Other challenges included changes in course instructor, process model (spiral to risk-driven waterfall), and documentation approach (laissez-faire to put-everything-on-the-Web). There were also infrastructure surprises: the Integrated Library System's server and search engine, which we expected to be available for Cycle 3, were not.

Risk management. Despite these obstacles, each project successfully delivered its IOC package—code, life-

cycle documentation, and demonstrations—on time. We believe a major reason was our strong emphasis on risk management, which enabled teams to depart from a pure waterfall approach to resolve whatever critical risk items surfaced. We had each team form a top-*n* risk list, which helped them characterize each cycle and gave everyone a flavor of what to expect.

The risk list helped the team prioritize risks by assessing risk exposure (probability of loss times magnitude of loss). Each week, the team reassessed the risk to see if its priority had changed or to determine how much progress had been made in resolving it. A key strategy was design to schedule, in which a team identified a feasible core capability and optional features to be implemented as the schedule permitted.

Some risks from a typical team risk list included

- *Tight schedule*. Risk aversion options included studying the requirements carefully so as not to overcommit, descoping good-to-have features if possible, and concentrating on core capabilities. Risk monitoring activities included closely monitoring all activities to ensure that schedules are met.
- *Project size*. Risk aversion options included descoping good-to-have features and capabilities if requirements were too excessive and identifying the core capabilities to be built.
- *Finding a search engine*. Risk aversion options included conducting a software evaluation of search engines, actively sourcing free search engines for evaluation and selection, and determining the best one for the project. Risk monitoring activities included submitting evaluation reports and conducting demonstrations so that an informed decision can be made.
- *Required technical expertise lacking*. Risk aversion options included identifying the critical and most difficult technical areas of the project and having team members look into them as soon as possible. Monitoring activities included closely following the progress of critical problems and seeking help if necessary.

Client involvement and reaction. The librarians' involvement with the student teams during the second semester was, for the most part, qualitatively and quantitatively different than during the first semester. Major system requirements had already been negotiated, but there were a few new requirements that added subtle differences to the original concepts. Nonetheless, the time required for the librarians' participation was not as extensive as it had been.

Except for one project, the librarians were delighted with the final presentations. Five library clients wanted either to adopt the application as is or extend it for possible adoption. The sixth application—the only one not well received—was an attempt to integrate the three photographic-image projects (stereoscopic slides, Hancock Library photo archive, Los Angeles regional history photos) into a single application. The team had only a short time to patch together pieces of three architectures and user interfaces. Some resulting features were good (a colored-glasses stereo capability with good resolution, for example), but none of the clients were enthusiastic about implementing the results.

The librarians said that working with Theory W and the WinWin philosophy made it easy for them to "think big" about their projects.

The librarians expressed that working with Theory W and the WinWin philosophy made it easy for them to "think big" about their projects. The negotiation process balanced that vision by allowing teams and librarians to agree on a feasible set of deliverables for the final products during the academic session. And, although the time commitment was not great, participation in this project let the librarians focus part of their time on multimedia applications and software engineering. One of the greatest advantages for librarians was that they became more familiar with digital library issues and the software engineering techniques involved in their implementation.

Nature of products. As one librarian noted in her evaluation memo

> The interaction between the student teams and the librarians produced obvious differences in products designed for different users. For example, the technical reports interface mirrored the technical nature of the type of material included and expected future users of the system, while the student film archive interface reflected the needs and interests of a very different clientele.

Figure 6, the user interface for the medieval manuscripts application, typifies the look and feel of the products. The Netscape-based application uses various windows to display the manuscript's attributes, to query for desired manuscripts using a search engine, and to enter and catalog new manuscripts.

Adoption of applications. The students spent summer 1997 refining two of the five applications. However, only one of these—the student film archive—was actually implemented. As it turned out, this application was the only one with sufficient budget, people, and facilities to sustain the product after it was implemented. In the following year, we agreed to let the USC library choose which applications would be developed to IOC in spring 1998, and we agreed that we would implement only the applications the client could sustain.

LESSONS LEARNED

When we started the course, we were not sure about any of our choices on such issues as team size, docu-

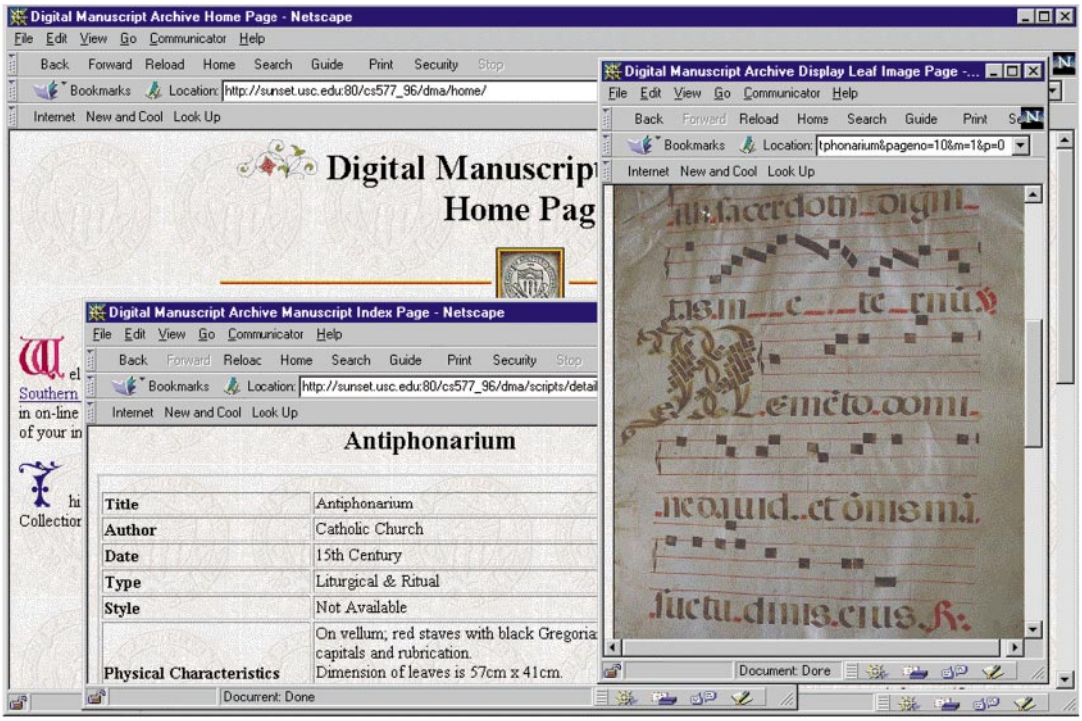


Figure 6. The user interface for the medieval manuscripts application in Figure 3a. The application satisfies the client's need to scan medieval manuscripts in a way that permits researchers to simultaneously study special markings and read historical data about the image.

ment guidelines, tools, milestones, and course material. However, the library clients and management found the projects sufficiently valuable that they committed both to a continued series of similar projects and to supporting the product's transition and sustaining it after implementation. We, in turn, obtained extensive data and feedback on how to improve the course and project approach both for future courses and for industrial practice.

Number of cycles. For projects of this size, using a single cycle each for the LCO and LCA milestones was about right. Smaller projects can get to the LCA milestone in a single cycle; larger projects may take several cycles to achieve their LCO and LCA goals. Given the results of our LCO reviews, using a single cycle would have produced less satisfactory results in about half the projects. In several projects, the detail was not balanced in either the archiving or query/browsing parts of the LCO packages; the LCA cycle let them correct that imbalance. Using three cycles to produce the LCO and LCA milestones would have left insufficient time to both produce and coordinate three sets of artifacts.

Degree of flexibility. The teams were able to adapt to real-world conditions, such as pleasant and unpleasant surprises with COTS packages, the unavailability of expected infrastructure packages such as the server, lack of expertise on library information systems; and personnel complications. More formal or contract-oriented approaches would not have been able to accommodate these changes in the short time (11 weeks) available.

Communication and trust. We found that, at least for this type of application, the most important outcome of product definition is not a rigorous specification, but a team of stakeholders with enough trust and shared vision to adapt effectively to unexpected changes. In the beginning, the library clients were considerably uncertain about going forward with the projects. By the LCA milestone, however, the uncertainty and doubt about working with the student teams had been replaced with enthusiasm and considerable trust, although many were still uncertain about the applications' technical parameters. This growth continued through the development period and led to a mutual commitment to pursue additional projects in the following year. The ability of the WinWin approach to foster trust was consistent with earlier experiences.[6]

Smooth transitions. In previous uses of the WinWin spiral model, the transition from WinWin stakeholder agreements to requirements specifications had been rough. The WinWin groupware tool helped smooth this transition. Mapping the WinWin domain taxonomy onto the table of contents of the requirements specification and requiring the use of the domain taxonomy as a checklist for developing WinWin agreements effectively focused stakeholder negotiations. We are exploring how to automate parts of the requirements transition to make it even smoother.

Use of developer time. Although our approach avoided some inefficiencies, we still experienced significant bottlenecks from documentation overkill and attempts to coordinate multiple views. The second-year projects (described later) had less redundant and voluminous documentation, used the Rational Rose integrated object-oriented tool set (which decreased the amount of documentation), and thus yielded fewer inconsistencies. We also had five instead of six members per team, which reduced inconsistencies and overhead because fewer people had to talk to one another.

Finally, we added training and opportunities for feedback. The WinWin groupware tool helped with team building and feature prioritization, but people needed

more preliminary training and experience in its use. Students also cited the need for more training on key Web skills and more feedback on intermediate products. For the second-year projects, we added homework examples both on WinWin principles and preliminary use. We set up special sessions for training on WinWin and Web prototyping. We also set up special LCO and LCA architectural review board sessions for all projects, rather than just three in-class sessions.

Client acceptance. We learned two lessons here. The first is don't finish negotiations before prototyping. If you do, the agreements destabilize once the clients see the prototypes. In the second-year projects, we had the teams negotiate and prototype concurrently. The second lesson is make sure the clients are empowered to support the product not just with knowledge and enthusiasm, but also with resources for the product's operation and maintenance. In the second-year projects, this became our top criterion for selecting applications.

RESULTS OF SECOND-YEAR PROJECTS

In the second-year projects, which we just completed, 16 teams developed similar library-related projects. The process of the second year followed the process of the first year for the most part (from the LCO to LCA to IOC milestones). The changes we made reflected customer and student wishes, their suggestions, and other lessons learned during the first year.

From the 16 projects in the first semester, the clients selected five applications for development according to the library's commitment to sustain them after the second semester (IOC). Four are now transitioning to library operations, and the fifth has good prospects for transition after refinement this summer. We adapted several parts of the first-year process in the second-year projects.

Documentation. We restructured the document guidelines to reduce duplication, and to adapt them for use with Rational Rose and the Unified Modeling Language (we had used OO development and design methods the first year but we did not provide particular tool support for it). The average length of the LCO package decreased from 160 pages in the first-year projects to 103 in the second year.

Layered architectural description. Using UML and Integrated Systems Development Methodology (ISDM)[7] as our object-oriented methods, we were able to more strongly refine our system software architectural description into three model layers: domain description, system analysis, and system design. Each layer was analogous to the others, but had a different intended audience. The layering improved internal consistency because it maintained distinct relations (documented via simple reference tracing) between the views within and outside each layer. Many teams still found the concepts of consistency and tracing difficult to grasp, but they were more aware than in the first-year projects that these issues were important. Through the domain description, the teams were able to rapidly understand the parts of their client's domain that were relevant to the target system. With this intermediate representation, the teams were able to work with the client to communicate vital responsibilities, qualities, and components of the target system without losing the client in too much technical design detail. During system analysis, one client commented "I can really see that this [the system] has all the things I expected and is what I wanted." In all, layering helped manage the architectural complexity by letting teams capture, validate, and refine information in a practical and useful way as well as communicate them effectively.

Client acceptance. The extra WinWin and prototyping preparation and training, early prototyping, and adoption of LCO and LCA review boards fit naturally into the WinWin spiral approach and increased the productiveness and quality of the second-year projects over the first-year efforts. There were fewer requirements breakdowns in the later stages of the life cycle, which increased client participation and acceptance to the point of "client activism." Indeed, when a team was given some criticism by the review board, often the client would actively defend the team and their efforts. The resulting discussions often led to identifying additional important objectives, constraints, and alternatives, making the spiral model iterations more effective, and producing more satisfactory products for the clients. The overall satisfaction rating from client critiques, on a scale of 1 to 5, went from 4.3 in the first year to 4.7 in the second.

We found that the most important outcome of product definition is not a rigorous specification, but a team of stakeholders with enough trust and shared vision to adapt effectively to unexpected changes.

We are currently addressing improvements for the third year of projects. Of the 80 students in the second-year projects, 26 indicated the need for more UML and Rose education (18 indicated that UML and Rose were very helpful), 13 indicated the need for better document guidelines, and nine indicated the need for a Cocomo II model calibrated to the student projects.

We believe our results so far indicate that the WinWin spiral model will transition well to industry use. The digital library projects were in a sense an industry test because about 20 percent of the teams were purely industry employees, and additional teams had mixes of industry employees and full-time students. In fact, since the first-year projects, industrial organizations have adopted many elements of the WinWin spiral model. Rational, for example, has adopted the LCO, LCA, and IOC definitions as the major milestones in their Objectory or Rational Unified Management Process.[8,9] MCC is developing an industrial-grade version of the WinWin tool as part of its Software and System

We believe our results so far indicate that the WinWin spiral model will transition well to industry use.

Engineering Productivity project. Several other USC-CSE affiliate organizations are adopting WinWin spiral model concepts into their lifecycle process approaches. TRW has also successfully scaled up the model in its CCPDS-R project,[9] in which it delivered a million-line command and control system within budget and schedule—an effort considered a showcase project by its Air Force client.

New software process models generally take years to validate. The original spiral model was created in 1978, first tried on a 15-person internal project in 1980, scaled up to a 100-person contract project in 1988, and became a fully documented method in 1994. For the WinWin spiral model, we were fortunate to find this family of multimedia applications that has let us continue validation and improvement on a one-year cycle. As we refine the model and its attendant processes, and as more of our industry affiliates begin using the model, we hope to amass even more concrete validation data.❖

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: Allied Signal, Bellcore, Boeing, Electronic Data Systems, Federal Aviation Administration, 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, 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 Denise Bedford, Anne Curran, Simei Du, Ellis Horowitz, Ming June Lee, Phil Reese, Bill Scheding, and Nirat Shah for support in key areas.

References

1. B. Boehm and R. Ross, "Theory W Software Project Management: Principles and Examples," *IEEE Trans. Software Eng.*, July 1989, pp. 902-916.
2. B. Boehm et al., "Cost Models for Future Software Processes: COCOMO 2.0," *Annals Software Eng.*, Vol. 1, 1995, pp. 57-94.
3. B. Boehm, "Software Risk Management: Principles and Practices," *IEEE Software*, Jan. 1991, pp. 32-41.
4. I. Sommerville, *Software Engineering*, 5th ed., Addison-Wesley, Reading, Mass., 1996.
5. G. Booch, I. Jacobson, and J. Rumbaugh, "The Unified Modeling Language for Object-Oriented Development," Ver. 1.0, Rational Software Corp., Santa Clara, Calif., 1997.
6. B. Boehm and P. Bose, "A Collaborative Spiral Software Process Model Based on Theory W," *Proc. Int'l Conf. Software Process*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 59-68.
7. D. Port, "Integrated Systems Development Methodology," Telos Press, 1998 (to appear).
8. "Rational Objectory Process," Ver. 4.1, Rational Software Corp., Santa Clara, Calif., 1997.
9. W.E.Royce, *Unified Software Management*, Addison-Wesley, Reading, Mass., 1998 (to be published).

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 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 USC's Center for Software Engineering. His research interests are in software architecture and requirements negotiation. He received an MS in computer science from USC and is a student member of the IEEE.

Julie Kwan is executive and research programs librarian for the Marshall School of Business, Information Services Division at USC. Her interests include information needs and information-seeking behaviors; business, scientific, and technical information transfer; and customer-analysis methodologies. She received an MS in library science from the University of Illinois and is a member of the American Library Association, the Medical Library Association, and the Special Libraries Association.

Dan Port is a research assistant professor at USC and a research associate with the Center for Software Engineering. His primary research interests are in component and object-oriented architectures, systems integration, and partially ordered event structures. He received a PhD in applied mathematics at the Massachusetts Institute of Technology.

Ray Madachy is the manager of the Software Engineering Process Group at Litton Guidance and Control Systems and an adjunct assistant professor of computer science at USC. He received a PhD in industrial and systems engineering from USC. He is a member of the IEEE, ACM, and the International Council in Systems Engineering.

Contact the authors through Egyed at the Center for Software Engineering, USC, Los Angeles, CA 90089-0781; aegyed@sunset.usc.edu.