DEMONSTRATION GUIDE

USC Center for Software Engineering
Department of Computer Science
Research Review
March 11-12, 1996
# Demonstration Schedule

**Monday, March 11**  
**Time:** 4:30-6:00pm

<table>
<thead>
<tr>
<th>Demo</th>
<th>Demoer</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinWin Hands On (1)</td>
<td>Ellis Horowitz</td>
<td>SAL 324</td>
</tr>
<tr>
<td>QARCC and S-COST</td>
<td>Hoh In</td>
<td>SAL 320</td>
</tr>
<tr>
<td>Strikeware Application of WinWin</td>
<td>Barry Boehm, Prasanta Bose, Frank Belz, Judy Kerner</td>
<td>SAL 324</td>
</tr>
<tr>
<td>WinWin Hands On (2)</td>
<td>Ming June Lee, June Sup Lee, Judy Kerner</td>
<td>SAL 324</td>
</tr>
</tbody>
</table>

**Tuesday, March 12**  
**Time:** 12:10 - 2:00 pm

<table>
<thead>
<tr>
<th>Demo</th>
<th>Demoer</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinWin</td>
<td>Ellis Horowitz</td>
<td>SAL 324</td>
</tr>
<tr>
<td>QARCC and S-COST</td>
<td>Hoh In</td>
<td>SAL 320</td>
</tr>
<tr>
<td>Architect's Automated Assistant</td>
<td>Ahmed A. Abd-Allah</td>
<td>SAL 327</td>
</tr>
<tr>
<td>USC COCOMO 2.0</td>
<td>Brad Clark, Chris Abts, Suniza Devnani</td>
<td>SAL 324</td>
</tr>
<tr>
<td>I-Doc</td>
<td>Lewis Johnson, Ali Erdem</td>
<td>SAL 200C</td>
</tr>
<tr>
<td>Internet-based Process Support</td>
<td>Walt Scacchi, John Knoll</td>
<td>SAL 200C</td>
</tr>
</tbody>
</table>
Demonstration Description

Name: WinWin: A Support System for Collaboration and Negotiation of Requirements

Objective: Develop support environment for the WinWin Spiral Process model - that aids in the capture, cooperation and negotiation of multi stakeholder requirements for a large systems.

Rationale: Current process models and associated support environments fail to involve the multiple stakeholders whose concerns are critical to the success of a project. The WinWin Spiral process model addresses this problem by considering the stakeholder's win conditions as fundamental to all decisions.

Target Users: Initially all participants in the front end of the software process: system engineers, representatives of users, customers, developers, maintainers.

Scope: WinWin provides an environment in which stakeholders can define and refine win conditions for the next step of the Spiral Model. There is support for collaborative product and process definition and negotiation. The environment offers tools for analyzing conflicting win conditions.

Project Type: Multi-year USC-CSE research project. Supported by ARPA and USC-CSE Affiliates.

Developers: Development team includes Anne Curran, Eul Gye, Ming June Lee, and June Sup Lee. The project leaders are Barry Boehm, Prasanta Bose and Ellis Horowitz.

Runs On: Sun hardware platform running X11 server; no other restrictions.

IPR Status: WinWin copyright owned by USC-CSE.

Technical Approach: The WinWin support system has a distributed architecture. A stakeholder interacts with a WinWin client that communicates to other WinWin clients via WinWin router called the DB-server. The DB-server maintains information on the stakeholders involved in a project and their corresponding WinWin database locations. Each WinWin client makes uses of a cache having a copy of the WinWin DB of other clients. It uses the copy for local queries and supporting the stakeholder activities. A set of tools are directly integrated with the WinWin - for example the COCOMO tool for cost analysis, the Netscape and Frame-maker tools for documentation and printing. The support environment provides a basis set of operations on the WinWin artifacts (the Win conditions, Issues, options and agreements) such as relating, commenting, state propagation, and messages.

Future Directions: Architecture-level decision rationale capture and use in collaborative design. Tools for handling conflicts, risks and uncertainties. Integration with other tools.
Name: AAA: Automated Architect’s Assistant

Developers: Ahmed A. Abd-Allah (aabdalla@cs.usc.edu)

Objective: Aid in the specification and analysis of software system architectures, their composition, and the resulting mismatches

Target Users: Software system architects

Scope: Large software systems which incorporate many subsystems of different styles, or which are based on or different COTS packages

Runs On: SunOS 4.1.3

Description: The Architect’s Automated Assistant (AAA) is a computer program which allows the specification and analysis of software system architectures. Software system architects can use this tool to specify large system architectures such as for a satellite ground station or a complex software engineering environment. These specifications can be subsequently analyzed by the tool for architectural mismatches that can be prohibitively expensive to correct if left till later stages in the lifecycle.

AAA revolves around the concept of architectural styles. Each system specified in AAA is expressed as an instantiation of a particular architectural style. Examples of architectural styles include main/subroutine, pipe & filter, distributed processes, and event based. The notion of style can also be used to capture COTS packages, where each package presents its own particular style.

Architects can also specify how subsystems are composed into larger systems using AAA. The tool will check the composition and the underlying subsystems for consistency with each other, reporting any constraint conflicts or mismatches. AAA internally supports constraints on different architectural views to varying degrees, including structure, topology, behavior, and interoperability.

Technical Approach: The tool is designed around a state-based formal specification of architectural styles and their composition. The specification has been done in Z, and the implementation has been done in Prolog.

Future Directions: Extend the formal specification to better support different architectural views and analyses, and correspondingly extend the tool.
Name: USC COCOMO v2.6

Objective: Develop a prototype software implementation for the COCOMO 2.0 Post-Architecture Cost Estimation Model.

Rationale: Software development practices have changed since the introduction of the COCOMO 1981 model. This new model estimates Software Development Effort and Schedule based on current and projected development practices.

Target Users: Project planners and managers.

Scope: This tool is used in the project planning and tracking activities during software development. Product, project, personnel, and platform information is entered about each software subsystem under development. Effort for each subsystem is computed. Effort and schedule for the entire system is also given.

Project Type: USC-CSE Research Project

Developers: This latest release was created by the team of Thomas Majchrowski, Lloyd Manglapus, and Suppachai Sangtongkhamsuk. The project leader was Professor Horowitz. This version is based on previous USC COCOMO software created by many others.

Runs On: PC with Windows 3.11 and Sun UNIX workstations.

IPR Status: Copyright to USC-CSE.

Technical Approach: The COCOMO 2.0 Model was proposed in July 1994. It supports estimation for two types of software development: Applications Composition, or one of System Integration, Application Generation, or Infrastructure type development. The model takes as input a number of factors that characterize the software product, the personnel, the development platform, and the project. These factors are used to adjust the size of the software product and produce an effort estimate.

Future Directions: In May of 1995 the Model and associated research were reviewed by the CSE Affiliates. Since then, CSE has been gathering data from Affiliates for use in calibrating the model. A Beta release of the Model to Affiliates is expected soon. The first release to the public is expected in the Fall of 1996. Research continues on modeling the effects of Commercial off the Shelf (COTS) software and the estimation of quality factors.
Name: QARCC (Quality Attribute and Risk Conflict Consultant)

Objective: One of the biggest risks in software requirements engineering is the risk of overemphasizing one quality attribute requirement (e.g., performance) at the expense of others at least as important (e.g., evolvability and portability). This exploratory knowledge-based tool is for identifying potential conflicts among quality attributes early in the software/system life cycle.

Rationale: Quite a few software projects have failed because they had a poor set of quality attribute requirements, even though they may have had well-specified functional and interface requirements. An important step in achieving successful software requirements and products is to achieve the right balance of quality attribute requirements.

Target Users: Initially all participants in the front end of the software process: system engineers, representatives of users, customers, developers, maintainers.

Scope: Aids for identifying conflicts among 7 top-level quality attribute requirements.

Project Type: Multi-year USC-CSE research project.

Developers: Developed by Hoh In. The Project leader is Barry Boehm

Runs On: Sun hardware platform running X11 server; no other restrictions.

IPR Status: QARCC copyright owned by USC-CSE.

Technical Approach: QARCC examines the quality attribute tradeoffs involved in Software Architecture and Process Strategies (e.g., one can improve portability via a layered architecture, but usually at some cost in performance). It operates in the context of the USC-CSE WinWin system, a groupware support system for determining software and system requirements as negotiated win conditions.

Future Directions: Sharpen knowledge base to provide fewer false alarms and more situation-specific advice. Extend tool to address lowest-level conflicts (e.g., security vs. fault-tolerance).
Name: S-COST (Software Cost Option Strategy Tool)

Objective: This exploratory knowledge-based tool is for assisting stakeholders to surface appropriate cost-resolution options, to visualize the options, and to negotiate a mutually satisfactory balance of requirements and cost.

Rationale: A critical success factor in requirements engineering is the balance of requirements commitments with available cost resources. Many software systems have come to grief due to a combination of commitments to costly but low-utility requirements (gold-plating), and/or inadequate budgets for high-utility requirements.

Target Users: Initially all participants in the front end of the software process: system engineers, representatives of users, customers, developers, maintainers.

Scope: Option generation and negotiation aids for top-level cost and quality conflicts.

Project Type: Multi-year USC-CSE research project.

Developers: Developed by Hoh In. The Project leader is Barry Boehm

Runs On: Sun hardware platform running X11 server; no other restrictions.

IPR Status: S-COST copyright owned by USC-CSE.

Technical Approach: S-COST examines the COCOMO cost drivers and Win Conditions and provides draft cost Options using the Cost Resolution Strategies (e.g., Reduce/defer functionality, Reduce/defer quality, Improve tools, techniques, platform, personnel capability, Relax schedule constraint, Reuse software assets, and Increase budget) with visualization aids. It operates in the context of the USC-CSE WinWin system, COCOMO, and QARCC.

Future Directions: Development of more detailed cost-related option generation and resolution capabilities.
Objective: Develop a software explanation system that will dynamically produce documentation tailored to the user's task and expertise to make the software understanding easier for the user.

Rationale: Software engineers, especially maintainers, spend a lot of their time trying to understand the software. Documentation is frequently not used in this process, because there is a high cost of search to find the answers in the documentation if the documentation is up-to-date at all. Recent studies have shown that minimalist manuals that are tailored to the user task improves productivity significantly. We are attacking the cost of search, update and understandability problems of the documentation in Intelligent Documentation project.

Target Users: Software developers, maintainers and other personnel

Scope: This tool is used in both authoring and viewing documentation. It makes it easy for the users to store their discoveries about the software as annotations. It includes a knowledge representation language and a natural language generator for building finer grain representations than annotations. The information in the knowledge base is used in generating documentation and answering user queries.

Project Type: ARPA Research Project

Developers: This latest release was created by the team of Ali Erdem, Rogelio Ada-bati and Amy Biermann. The project Leader was Lewis Johnson.

Runs On: A Unix platform such as a Sun Sparcstation 20 with at least 64 MB of memory is recommended. The presentation system and server also run on Silicon Graphics workstations. The presentation system and the software repository may be run either on the same workstation or on separate workstations connected over the Internet.

Software requirements: Requires Software Refinery's Refine/Ada and Allegro Common Lisp 4.2. I-Doc employs common Web browsers (NCSA Mosaic or Netscape), and the NCSA HTTPD Web server. A Perl interpreter is required on the server machine.

IPR Status: Copyright to USC.

Technical Approach: This demonstration will illustrate the use of the I-Doc dynamic documentation system, an information tool for software engineers. The tool dynamically selects, organizes, and presents information about a software system, to help answer specific questions that software engineers might have. The system actively seeks to explain those aspects of the system of interest to the engineer. This greatly reduces the amount of effort spent in understanding software, a critical concern for software maintenance and reengineering. The system will be demonstrated on military application codes written in Ada.
I-Doc consists of three components: a software repository, a presentation generator, and a hypertext viewer. The software repository contains annotated source code, design components, and domain concepts, sufficient to provide answers to a range of types of questions. The presentation generator inputs information from the user about the task that they are trying to perform and the questions that they are trying to answer, sends queries to the software repository for relevant information, composes hypertext presentations, and sends them to the viewer. The viewer provides the user interface both for inputting questions and presenting answers. The system is built on top of World Wide Web browsers and servers, minimizing the amount of special-purpose software required in order to use the system.

**Future Directions:** An initial version of I-Doc is available at [http://www.isi.edu/fso/i-doc.html](http://www.isi.edu/fso/i-doc.html). We are working on enhancing the knowledge base for providing higher levels of abstractions for the user. We are also working on the natural language generation and the graphical presentation issues.