Evolutionary Design of Complex Software (EDCS) Demonstration Days 1999

Wayne Stidolph, ed.

http://www.if.afrl.af.mil/programs/edcs/demo-days-99/Index-By-Technology.html

Abstract

This report summarizes the Product/Technology demonstrations given at Defense Advanced Research Projects Agency (DARPA) Evolutionary Design of Complex Software (EDCS) Program Demonstration Days, held 28-29 June 1999 at the Sheraton National Hotel, Arlington, VA.

The Evolutionary Design of Complex Software (EDCS) Program

Evolutionary Systems are those that are capable of accommodating change over an extended system lifetime with reduced risk and cost/schedule impact. Most of our complex defense systems depend on software for their successful operation and, by its very nature, the software in those systems is the primary vehicle for adapting systems to change.

The ultimate vision is to evolve software systems faster, better, and cheaper -- through automated change. The EDCS Program is providing for the development and experimental application of leading-edge software technologies, which can achieve that vision, and enable significant improvements in military mission effectiveness and information superiority. The goal is the capability to produce software intensive military systems that are highly flexible and adaptable to meet changing requirements -- evolutionary systems.

Desert Storm clearly demonstrated the need for great flexibility in the just-in-time use of superior technology that can be provided by software. In the post cold war era, the exact nature of potential threats is difficult to assess. Trends toward downsizing and budget reduction place an increased burden on systems to last longer and to adapt quickly to change - changes to mission, changes to hardware/software and system configuration, changes to operational elements, and changes in volume/types of data processed.

Today's needs for flexibility and adaptability in military systems transcend the conventional notions - we need dynamic flexibility and dynamic adaptability. Systems must evolve and adapt during development and deployment, and throughout their extended operational life.

Phase I of the EDCS Program has some exciting demonstrations available which illustrate how the emerging technologies it is producing can apply to a number of military scenarios related to evolving software for the warfighter, as well as to many commercial applications.

Index of Demonstrations

- Acme and AcmeStudio
- Adaptation and Commitment Technology (ACT)
- Arabica – JavaBeans Composition Tool
- Architectural Analysis of Component-Based Systems
- ArchStudio – Architecture-Level Runtime Evolution
- Argo/UML – Object-Oriented Design Tool
- ARGUS-I: "All-Seeing" Architectural Analysis
- AverStar EWatch(tm)
- Capabilility Packages for Avionics Software (CPAS)
- CHIME - framework for immersive 3D collaborative virtual environments
- Chimera – Open Hypermedia System
- CodeSurfer – Dependence Graphs and Program Slicing
- Colorado Distributed Software Engineering Technologies - Software Dock, SRM, DVS, Siena, Architecture, Hypermedia/Databases
- Complex Event Processing (CEP)
- Composition of Multi-site Software (CHAIMS)
- COTS based Design Editor for User Specified Domains
- DAS-BOOT: Design-, Arch- and Spec-based approaches to OO Testing
- Demeter/Adaptive Programming
- DoME – Domain Modeling Environment
- Endeavors – Open, Distributed, Extensible Workflow Support Environment
- Esprit de Corps Suite of tools that support the MORALE legacy software evolution method - ISVis, SIRRINE, VisEd, SAAMPad, ACMEServer, REMORA, MORPH, ScenIC View, Dowser
- Expectation-Driven Event Monitoring
- FLAVERS – Program Behavior/Property Verification System
- Formal Alternative Management Integrating Logical Inference and Rationales (FAMILIAR)
- Incremental Constraint Engine
- INSERT - Incremental Software Evolution for Real-Time Systems
- Internet-Based Information Management Technology – Worklets, Workgroup Cache, Xanth, TreatyMaker, JPernLite, TaskWeb
- Jakarta Tool Suite (JTS)
- Knowledge Depot – Project Awareness
- Little-JIL – Graphical Specification Language for Executable Processes
- Maude – ADL Interoperability
- MBASE - Model-Based (Systems) Architecting and Software Engineering
- MediaDoc: Automated Generation of Multimedia Explanatory Presentations
- MetaH – Architecture Description Language
- Model Integrated Computing (MIC)
- ORBIT/VIRTUE - Collaboration and Visualization Support for Complex Systems Evolution
- Oregon High Assurance Technology
- SAAGE - Software Architecture, Analysis, Generation, and Evolution
- Quest – Software Analysis and Testing
- Securely Wrapping COTS Products
- Siddhartha - Specification-Based Testing of Low-Testability Programs
- SoBeIt: Structural and Behavioral Execution Instrumentation Tool
- Software Composition Workbench
- Specware – Specification and Design System
- TestTalk – Test Description Language
- UML/Analyzer - A System for Defining and Analyzing the Conceptual Integrity of UML Models
- WebDAV – Distributed Authoring/Versioning over the WWW
- Carnegie Mellon University (Composable Systems Group) – David Garlan

Demonstration Summaries

Acme and AcmeStudio: http://www.cs.cmu.edu/~Compose/
As part of the Acme and ADL Toolkit projects, we have developed tools and libraries to support the integration of different ADL technologies developed within the EDCS community. Our ultimate goal is to lay the foundation necessary for assembling integrated environments from the diverse set of independent tools developed within the community. Our solution is to provide a shared representation, the Acme language, guidance on translation to and from Acme, a library to support that effort, AcmeLib, and finally, a customizable graphical editing and analysis environment, AcmeStudio, through which tools may be accessed.

We will demonstrate some of the customizable features of AcmeStudio and show how this infrastructure can be adapted to different design domains and its analysis capabilities extended using Acme and the AcmeLib class library to develop new tools and integrate existing tools. We will also demonstrate how we used the Armani language and its constraint-language extensions to Acme as part of this effort.

Adaptation and Commitment Technology (ACT): http://www.cs.cmu.edu/~wls/act/
Carnegie Mellon University – William L. Scherlis

CMU will demonstrate a prototype program evolution tool for Java. The tool uses source-level program analysis, annotation, and manipulation techniques to support programmer-directed evolutionary change with a high level of assurance. A principal focus is on enabling structural changes, such as class-hierarchy reorganization, data representation change, code specialization, and other manipulations to reorganize code. These changes are typically risky and costly for programmers to make, since they ordinarily involve large numbers of detailed code-level operations distributed through the text of a program, and they can be hard to validate.

The tool supports "99% pure" Java (with the remainder pending resolution of outstanding language design issues in the Java community). It embodies a growing number of techniques for source-level program analysis, annotation, and manipulation. The annotation system enables significant evolutionary changes to be made even when only portions of a system are available for analysis, as is the case for most distributed component-oriented systems development, including most Java code.

Although the tool embodies a number of sophisticated program analyses and manipulations and it assures correctness of the changes it makes, the programmer interface is nonetheless straightforward and intuitive -- many of the manipulations can be carried out using simple drag-and-drop gestures.

When the programmer makes a gesture, the tool automatically invokes appropriate manipulations and c+arries out supporting analyses, possibly interacting with the programmer in the process.

The tool creates a detailed code-level design record that includes details of the evolutionary steps. The design record enables flexible exploration of design alternatives, as well as providing an audit trail for system assurance. We offer a hybrid approach to assurance: If the analyses and manipulations triggered by a programmer gesture succeed, the tool will certify a change. If the change is potentially unsafe, the tool can offer the option to authorize the change nonetheless. These vouched-for changes can then potentially be assured later using the tool or other techniques, or, in the case of high assurance systems, they can be prohibited by policy. The intent is to allow programmers to evolve and assure code in a flexible manner.

The tool infrastructure has several important features, particularly fine-grained versioning support for program evolution, flexible persistence, visualization of code and related structures, and (through the CSpace project) a number of collaborative features. In our experiments, we have represented, for example, 10,000 versions of a 500,000 node system. The infrastructure itself is language-independent. The overall intent of the infrastructure design is to: (1) provide more effective retention in the design record of "perishable" information ordinarily lost in traditional software development, and (2) provide a pragmatic approach to assurance, in which each evolutionary step is explicitly assured either by programmer or by tool, but with minimal intrusion into the interactive process, which may involve exploration of multiple design alternatives.

The demonstration will show how a programmer can use the tool to evolve an example of hard-wired ad hoc Java code into a reusable set of components with a more principled design, and this configuration then subsequently specialized to a range of particular uses, with the tool assuring correctness throughout.

Arabica: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

Arabica is a tool designed to guide developers in the composition of JavaBeans into applications adhering to the Component-Connector (C2) architectural style. The developer can use its visual interaction mechanisms to drag and drop beans into the main palette to create applications in the C2 style. The tool provides an interactive dialog that the developer uses to define the interface of the bean component in the context of the C2 architecture within which it is to be deployed. A Style Dialog provides guidance in adhering to C2 style rules and constraints. Style rules are checked and enforced as the beans are composed incrementally in the palette. A bean framework is also provided to aid designers in creating C2 compliant bean components from scratch. The tool runs on all Java Platforms.

Architectural Analysis of Component-Based Systems: http://www.sdl.sri.com/
SRI International- Victoria Stavridou

SRI International has developed technology to support generation and analysis of abstract architectural descriptions, in a variety of architecture description languages, from a complex, component-based system implementation.(By "component-based", we mean that the implementation is constructed from instances of validated component and connector structures selected from a repository and, perhaps, a comparatively small amount of additional code written for the specific application.) The architectural descriptions

are produced by successive applications of validated abstraction patterns, selected by the system analyst from a repository.

The demonstration will show how this technology can be used to produce architectural descriptions in Sadl, Acme, and (ultimately) other ADLs from the implemented system. It will also show how these descriptions are then analyzed using the ADLs' toolsets (e.g., PVS for demonstrating dependability properties of the Sadl description, and AcmeStudio for visualizing the architecture based on the Acme description).

ArchStudio: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

ArchStudio is a tool suite that supports the design, analysis, implementation, and runtime evolution of software systems at the architecture-level.

ArchStudio's support for runtime evolution allows mission-critical systems to be upgraded without shutting down and restarting them, thereby avoiding costly downtimes. Our booth will showcase the versatility of our approach and technology by demonstrating dynamic changes to a logistics planning system for cargo routing. Without exiting and restarting the system, we are able to add more informative graphical depictions of the cargo routing process and incorporate an intelligent planning component that improves routing efficiency. Our technology has the promise of significantly reducing the costs and risks associated with evolving mission-critical systems.

Argo/UML: http://www.ArgoUML.com/
University of California, Irvine – Richard Taylor/David Redmiles

Argo/UML is an object-oriented design tool that helps designers make better design decisions. This demo will focus on Argo/UML's cognitive support features:

- Design Critics point out errors and incompleteness in the design as the designer works,
- Wizards help the designer correct identified problems,
- Smart Checklists improve on the paper-based checklists used by many organizations in design review meetings,
- Navigational Perspectives present specialized views of the design to support common design tasks,
- The Broom Alignment tool helps designers make neat looking diagrams without giving all responsibility to an automatic layout algorithm,
- Selection-Action Buttons put toolbar buttons for common actions directly around the selected diagram element leading to much easier diagram construction,
- Designer-Expert Communication Channels help connect people in the organization, and
- Design History shows which design changes caused or cured each identified problem.

Argo/UML also provides many standard CASE tool features, stores its data in standard XMI and PGML formats, and runs on all Java platforms. The tool currently supports the following diagram types:

- UML Class Diagrams
- UML State Diagrams
- UML Use Case Diagrams
- UML Activity Diagrams
- UML Collaboration Diagrams
- UML Sequence Diagrams (coming soon)
- UML Component Diagrams (coming soon)

Argo/UML is an active open-source project with over 100 members of its developers' mailing list and over 7000 registered users. Active areas of development include additional cognitive support, reverse engineering, enhanced XML support, and documentation generation. For more information or to download ArgoUML or take a visual tour, see http://www.ArgoUML.com/

ARGUS-I: "All-Seeing" Architectural Analysis: http://www.ics.uci.edu/~djr/edcs/PerpTest.html
Component-based software development relies fundamentally on the quality of the components of which a system is composed and there configuration, yet little technology exists for component-based quality assurance. To predictably and reliably build complex systems by composing components, components must be analyzed not only independently but also in the context of their connection to other components. Analysis should be coordinated, therefore, at a higher level of abstraction – e.g., at the software architecture level, where components, connectors, and their configuration are better understood and intellectually tractable.

ARGUS-I is a comprehensive set of specification-based analysis tools focusing on both the component and architecture levels. ARGUS-I facilitates iterative and evolvable analysis during architectural specification and implementation. Both structural and behavioral analyses are accomplished by a synergistic combination of static and dynamic techniques. While static analysis can, for instance, detect incompatibility between the data exchanged between components and verify architectural adherence to design heuristics and style rules, dynamic analysis may be required for revealing defects in the dynamic component interaction and communication behavior between components.

Component-level specification analyses statically verifies interface consistency and the relationship to the statechart behavior specification and dynamically evaluates component behavior through statechart simulation. Architecture-level specification analyses statically verifies structural and behavioral dependencies among components and dynamically evaluates the architecture configuration through architectural simulation.

Component-level dynamic analysis is performed by testing components to reveal inconsistencies between the implemented component and its statechart specification. Dynamic architecture analysis is based on architecture debugging, monitoring, and dynamic conformance verification between specification and implementation.

The current version of ARGUS-I works with software architectures in the C2-style but augments the C2 architecture description with component behavior specification described by Statecharts.

Ewatch: http://www.averstar.com/edcs
AverStar - Bill Carlson/Chris Garrity /

AverStar is presenting EWatch, a practical analysis and debugging tool for distributed, component-based software, with a particular emphasis on systems of components assembled dynamically both before and after deployment. The heart of EWatch is an extensible, event-driven framework that controls and accepts diagnostic inputs from software probes attached to the operational components. AverStar will be demonstrating the use of EWatch to monitor, analyze and debug systems of heterogeneous CORBA and COM components, and an innovative configuration control technology for managing the evolution of dynamic networks of distributed objects.

Capabilility Packages for Avionics Software (CPAS): http://www.northrop.com/cpas/
Northrop Grumman - Garry Brannum

Northrop Grumman demonstrated an Integrated Avionics Software Development Environment that supports evolutionary requirements negotiation, implementation,+ system test, and acceptance as part of last year's EDCS effort. That environment incorporates both in-house and EDCS developed technologies for software understanding, incremental test and certification, and architecture-driven system design and composition. Our demonstration scenarios focused on the B-2 and included both development and evolution of avionics software.

Northrop Grumman's 1999 Capability Packaging for Avionics Software (CPAS) program demonstrates the application of EDCS technology in the context of additional Northrop Grumman development programs. From these application exercises, we have identified development activities that benefit from the application of EDCS technology. These benefits include: (1) improved system specifications, (2) reduction in verification and test efforts, and (3) incremental validation of system updates. The CPAS99 demonstration scenario will illustrate the process required for limiting the impact of changes within incremental updates to new or evolving avionics systems. Process guidelines in two broadly applicable areas will be highlighted.

1) Formal Static Analysis of Source Code
- Improved approach for software understanding based on ontrol flow and data flow dependencies
- Impact assessment based on provable compartmentalization of the affects of system changes
- Formal selection of optimal regression tests based on overlapping dependency and validation sets

2) Event-Based Modeling, Abstraction, Analysis, and Validation
- Modeling and analysis of system behavioral specifications and architecture constraints
- Automated domain-specific abstraction of events from observable component interactions (bus messages, process activations, etc.)
- Validation of system behavior based on models (i.e. event models of expected behavior)

These two capabilities will be demonstrated in the context of:

1) A rapid communications-upgrade development for the B-2, to be included in the Air Force Expeditionary Force Experiment 1999 (EFX 99) prototype demonstration, and
2) The development and evaluation of avionics software for new aircraft.

Internet-Based Information Management Technology: http://www.psl.cs.columbia.edu/current.html
Columbia University, Dept. of Computer Science - Prof. Gail E. Kaiser

Columbia University will demonstrate its Internet-based information management technology for multi-organization collaborative work. Applications are not limited to very large systems engineering, e.g., open-source software and multiple (sub) contractor projects, but also include decision support and distance learning. The main components and toolkits include: Worklets, a mobile agents approach to meta-workflow for dynamic reconfiguration and knowledge propagation; Workgroup Cache, for zero-latency knowledge propagation among dynamically organized groups according to task-specific criteria; Xanth, an XML-based data fusion service; Groupspace Controller, an object/event broker featuring vetoable events and wraparound service activation; TreatyMaker, a toolkit for rapidly constructing and dynamically managing N-ary interoperable alliances among peer services and systems; JPernLite, transaction management middleware supporting plugin extended transaction models, e.g., for groupwork and "what if" transactions; and TaskWeb, an open hypermedia system for PDA's.

These and other technologies are integrated in CHIME, Columbia Hypermedia IMersion Environment, a framework for generating and managing MUD-like 3D virtual worlds for collaborative information understanding and interaction. Demo scenarios will include software development and multi-agency emergency response.

Chimera: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

Hypermedia is an effective technology for helping manage the myriad of heterogeneous artifacts, systems, and relationships, which exist in large-scale software engineering projects. Chimera is an open hypermedia system with explicit support for heterogeneity augmented by a deep integration with the WWW. Chimera provides a set of flexible abstractions, which enable the integration of hypermedia services into familiar tools. Integration with the WWW leverages the Web's robust support for distribution within Chimera, enabling remote access to Chimera's hypermedia information.

This demonstration will show Chimera operating on a number of third party applications (Abobe Acrobat, Adobe FrameMaker, Netscape, and Xemacs) which were integrated through different methodologies. This demo will also feature Chimera's port to different operating systems (NT and Linux).

CodeSurfer:
http://www.grammatech.com/products/codesurfer/codesurfer_index.html
GrammaTech - Tim Teitelbaum

GrammaTech is commercializing ten years of DARPA/ITO-sponsored research on dependence graphs and program slicing by Reps and Horwitz at the University of Wisconsin, and will demonstrate CodeSurfer, a software development, inspection, and maintenance tool based on that technology.

CodeSurfer builds a dependence-graph program representation and provides a GUI for exploring this web. The dependence graph includes forward and backward links between each assignment statement and possible uses of the values stored by that assignment. Pointer analysis is used so that indirect loads and stores through pointers are taken into account, as well as indirect function calls. Dataflow analysis is used so that links between unrelated assignments and uses are excluded. Operations that highlight forward and backward slices show the impact of a given statement on the rest of the program (forward slicing), and the impact of the rest of a program on a given statement (backward slicing). Operations that highlight paths between nodes in the dependence graph (chops) show ways in which the program points are interdependent (or independent).

CodeSurfer's scripting language (Scheme), which provides access to the dependence-graph program representation and the Tk widget set, is used for extensibility, for batch applications, and for implementing research prototypes of future software engineering/re-engineering tools. CodeSurfer is currently limited to ANSI C programs of less than 100K SLOC. GrammaTech is at work on a DARPA/ITO SBIR to extend the technology to C++ and Java, to make it scale up to bigger programs, and to explore its application to hardware description languages. Northrop-Grumman is developing a Jovial version of CodeSurfer.

Distributed Software Engineering:
http://www.cs.colorado.edu/users/serl/
University of Colorado - Dennis Heimbigner/Alexander Wolf

The University of Colorado SERL project will demonstrate technologies focusing on four areas of distributed software engineering:

- Distributed configuration and deployment of systems:
  - The Software Dock is a distributed, agent-based framework supporting software artifact deployment over a wide-area network.
  - The Software Release Manager (SRM) supports one-stop, web-based release and retrieval of interdependent software systems.
  - The Distributed Versioning System (DVS) provides federated configuration management.
- Event Notification:
  - Siena is an Internet-scale distributed event notification service allowing applications and people to send and receive notifications.
- Software Architecture:
  - Aladdin is a tool for analyzing inter-component dependencies in software architectures.
  - Menage is an architectural environment that adds the configuration concepts of variability, optionality, and evolution to architectural descriptions.
- Web-Based Hypermedia and Databases:
  - Chimera provides the infrastructure to enable hypermedia linking of information between heterogeneous applications.
  - The Web Integration Tool (WIT) applies federated database techniques to provide unified access to multiple Web data sources.

Complex Event Processing (CEP):
http://anna.stanford.edu/rapide/rapide.html
Stanford University - David Luckham

The Stanford Rapide project will demonstrate applications of Complex Event Processing (CEP) to commercial and DoD systems to address problems in the areas of:

- Intrusion detection and security: in particular, recognition of sophisticated attempts to gain access to protected resources in a system, or to attack the system to degrade its operational capabilities.
- Enterprise monitoring and management: applications of CEP to hierarchical monitoring of enterprise systems, performance monitoring, testing of conformance to design and architecture standards constraints, and usage policy constraints.

Composition of Multi-site Software (CHAIMS): http://www-db.stanford.edu/CHAIMS/
Stanford University Computer Science Department - Gio Wiederhold/Dorothea Beringer/Neal Sample/ Laurence Melloul

The CHAIMS project is developing a very high-level megaprogramming language to support composition of remote, heterogeneous software services. It is intended for applications as seen in logistics, where information and analyses require interaction with a variety of mostly independent and autonomous resources. As the use of the Internet spreads we expect a wide variety of value-added computational services to become available, complementing the pure informational and data resources.

The CHAIMS language CLAM is on a higher level than traditional languages, yet takes into account emerging complexity issues that arise when calling large, distributed services. The traditional CALL-statement is split up into substatements: pre-invocation setups, asynchronous invocation of services and extraction of results, and termination of services. Having several primitives also allows to add an additional primitive for pre-invocation estimates of the performance of a service - important for choosing optimal services at run-time, scheduling the order of invocations according to criteria like time or cost, and giving more control to the user during the execution of the megaprogram containing the composition. Similar objectives lead to a primitive for monitoring ongoing invocations. Note that the CHAIMS megaprogramming language only serves composition, not general programming tasks. Focusing on the issues critical to system integration will allow a high level of capabilities to be attained

without conflicting with important facilities provided by computational services written in traditional programming languages.

The CHAIMS system consists of two parts. For the persons providing megamodules there have to be tools that support them in wrapping legacy modules and presenting the necessary information about available megamodules to the persons using them. For the actual composition process, CHAIMS provides a compiler that not only compiles the megaprogram written in CLAM but also generates all the necessary client code for the access protocol CPAM which is layered on top of various distribution protocols like RMI, CORBA, DCE and DCOM, as appropriate to service providers.

Our approach is different from what is found in the commercial sector in that we try to address composition issues as they arise when composing large, heterogeneous and distributed services. Today, such composition is either not done at all (e.g. a person responsible for logistics does such composition manually by phoning various companies and entering the information manually into a spreadsheet) or it is done by hand coded systems that do not allow the dynamic creation of new compositions. Yet megamodules available for wrapping and integration are showing up in increasing numbers on the internet. Using these sources in a more automated and flexible way becomes now the challenge for the DoD and many organizations that want to take advantage of fast and flexible accesses to and composition of these services.

A COTS-based Design Editor for User Specified Domains: http://www.isi.edu/software-sciences/multi-gen/multi-gen.html

University of Southern California Information Sciences Institute (USC/ISI) - Bob Balzer

We've extended PowerPoint to enable user-defined semantics to be applied to diagrams. As diagrams are created or modified, external analysis programs can monitor and react to those changes. They can graphically annotate the diagram with errors, warnings, analysis results, or data collected from an executing instance of the diagram. They can also use the diagram changes as control commands to be imposed on the executing instance.

Multiple graphical domains will be demonstrated.

DAS-BOOT: Design-, Architecture- and Specification-based approaches to Object-Oriented Testing: http://www.ics.uci.edu/~djr/edcs/PerpTest.html

University of California, Irvine - Debra Richardson

DAS-BOOT is a prototype design - and specification-based testing tool for object-oriented systems. The current prototype has satisfied two early goals: (1) to define improved specification-based coverage criteria suitable for testing object-oriented software systems whose behavioral specification is modeled as a finite state machine (FSM); and (2) to develop techniques for generating test drivers and test cases with little interaction required by the human tester. To demonstrate these capabilities, DAS-BOOT currently supports testing Java based upon UML statechart diagrams (a widely accepted notation for requirements/design specification).

DAS-BOOT takes as input (1) a Java class to be tested, (2) a state-based specification of the class behavior using statecharts, and (3) a test coverage criterion; from this it produces as output test drivers embodying test oracles to automatically test the Java class against the behavioral specification.

Specification-based test coverage criteria are meant to define what to test based upon covering a model of the component under test. These criteria can be used in two very distinct ways. First, they can be used to test or simulate the model during the specification phase and thereby detect defects early in the software lifecycle. Second, the criteria can be used while testing the implementation to ensure that the behavioral model is adequately covered, thereby testing based upon what the component is required to do rather than merely what it actually does (as is indicated by code-based coverage criteria). DAS-BOOT supports FSM-based coverage criteria intended to measure how well a finite state model of software behavior has been covered, which can be used either to test the model itself or to test an implementation intended to reflect the model.

Using these FSM-based test criteria in the specification phase to test (or simulate) the model, is straight-forward, because the components upon which test cases and/or coverage measures are based are the same as the components being tested. On the other hand, using criteria based on the FSM model to test the implementation requires that the test requirements, which are based on the FSM model, be somehow associated with or mapped to the implemented objects under test. DAS-BOOT assists the developer/tester in making such associations and automates the actual testing process as much as possible by generating automated test drivers for each test case that not only force required coverage but also check to ensure that the implementation behaves according to the model.

DAS-BOOT is integrated into ARGUS-I as the dynamic component analysis tool.

Demeter/Adaptive Programming: http://www.ccs.neu.edu/research/demeter/

Northeastern University College of Computer Science - Karl J. Lieberherr (lieber@ccs.neu.edu)

Demeter uses aspectual decompositions to control tangling in designs and programs. The kinds of aspects we support are behavior, structure, navigation, synchronization, and distribution. We have also developed (but not yet fully implemented) a component model (Adaptive Plug-and-Play Components, AP&PCs) with interfaces and connectors that can describe any aspect.

Demeter applies a widely used principle, called the Law of Demeter, to normalize designs and programs for ease of evolution. Demeter uses a special purpose language for object navigation, called traversal strategies, to easily follow the Law of Demeter and to connect components.

Northeastern will demonstrate three tools: Demeter/Java, DJ and the AP Library.

- Demeter/Java is a Java implementation of AP technology. It is a powerful tool, but requires a high learning curve.

- DJ is the tiny sister of Demeter/Java that trades expressiveness for ease of applying the concepts of AP.
- The AP Library contains the (patented) core algorithms for compiling adaptive programs. Both Demeter/Java and DJ are clients of the AP Library. The AP Library is a part of the Architecture Tool Kit.

We will also explain the concept of AP&PCs and how they are useful for the design and implementation of complex systems. AP&PCs are a tool for aspect-oriented design and programming.

DoME: http://www.htc.honeywell.com/projects/dssa/
Honeywell Technology Center – Steve Vestal

The Domain Modeling Environment (DoME) is a toolset for building graphical modeling tools. DoME has been heavily influenced by its use to build tools for ADLs (e.g. MetaH and ControlH), and by its use to support multiple graphical ADLs within the same environment. A variety of scripting, data exchange, and prototyping capabilities are provided. DoME is a fairly mature and robust toolset and has been applied on over a dozen programs to create toolsets for several dozen graphical specification languages. DoME is available as "open source" software at http://www.htc.honeywell.com/dome.

Endeavors: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

Endeavors is an open, distributed, extensible workflow support environment. It improves coordination and management by allowing flexible definition, modeling, and execution of workflow applications. Endeavors combines a sophisticated process modeling language with features designed for easy customization by both technical and non-technical users. Endeavors uses a layered object model to provide for the object-oriented definition and specification of process artifacts, activities, and resources. Behavior of process objects is specified through the use of handlers: code invoked by the object in response to events received. Stored locally or loaded from a remote source, handlers are loaded and bound to objects at runtime, allowing them to be changed dynamically in the course of process execution. Handlers themselves may reflexively access the state of the workflow through Endeavors interfaces, allowing for analysis and optimization by components of the process itself.

We will demonstrate process development using the Endeavors graphical end-user process description language. This language features dynanism, which supports exception handling and evolution within the development, deployment, and evolution cycle. Pre-built process component libraries and WWW integration mechanisms will be used to illustrate important adoption issues in workflow.

Esprit de Corps Suite: http://www.cc.gatech.edu/morale/
Georgia Institute of Technology - Spencer Rugaber

Georgia Tech will demonstrate the Esprit de Corps Suite of tools that support the MORALE software evolution process. The tools include:

- ISVis - A reverse engineering tool which allows an analyst to extract an architecture by visualization of dynamic program event traces.
- SIRRINE - A design critiquing tool which suggests how to reconfigure a software system based on an example of how it fails to achieve a specified set of goals.
- VisEd - An interactive graphical editor and layout tool for building and displaying architectural descriptions written in the ACME interchange language.
- SAAMPad - A design rational capture and playback tool which supports the Software Architectural Analysis Method (SAAM) by providing automated capture and access of SAAM sessions to aid in documentation and design rationale understanding.
- ACMEServer - A server that manages access to ACME architectural descriptions allowing multiple, distributed tools to concurrently manipulate architectural information.
- REMORA - An environment for reconciling multiple representations of architectures. REMORA has advanced matching capabilities which automates combinations of architectural descriptions.
- MORPH - A user interface migration tool which supports the migration of a character-oriented, legacy application interface into a graphical user interface. Currently MORPH supports migration to tcl and to the web with HTML forms.
- ScenIC View - information systems which manages access and evolution of mission-oriented requirements information including scenarios and goal hierarchies.
- Dowser - Domain-oriented browsing of source code and documentation.

Expectation-Driven Event Monitoring: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

Expectation-Driven Event Monitoring (EDEM) can be used to shed light on how applications are used, to uncover mismatches in actual versus expected use, and to increase user involvement in the evolution of interactive systems. Software agents are deployed over the Internet to collect usage information to help developers make more informed design and effort allocation decisions. This demo focuses on how agents perform in-context data abstraction, selection, and reduction to allow meaningful information to be collected on a potentially large and ongoing basis over the Internet. Unlike traditional application instrumentation approaches, EDEM's architecture allows data collection to evolve flexibly over time without impacting application deployment or use.

FLAVERS: http://laser.cs.umass.edu/perptest/
University of Massachusetts at Amherst - Leon Osterweil/Lori A. Clarke

We will demonstrate FLAVERS a flexible, powerful system for automatically guaranteeing the absence, or detecting the presence, of a wide range of user-specified properties or behaviors. FLAVERS complements traditional testing approaches, which only demonstrate the presence or absence of errors for the specific test cases that have been executed. It also complements formal

verification methods, which employ more comprehensive analysis, but require a great deal of expertise on the part of the user. FLAVERS employs data flow analysis techniques that are efficient and easy to use and are applicable to both sequential and concurrent programs. With FLAVERS, users can specify a program behavior that is of particular importance and then direct FLAVERS to determine whether the behavior will occur on either all, some, or none of the program's executions. For example, FLAVERS can be used to verify safety conditions by demonstrating that a user-specified unsafe behavior cannot possibly occur. Or, it can help with debugging by identifying situations where a dangerous behavior can occur.

The FLAVERS demonstration will analyze programs written in Ada and Java and will highlight a new, more powerful and easy-to-use user interface. A key addition to this user interface is support for browsing paths along which errors can occur.

Formal Alternative Management Integrating Logical Inference and Rationales (FAMILIAR): http://www.kevol.com/KEI-6p.html
Knowledge Evolution/Synquiry Technologies - Dr. Sidney Bailin/Dr. Dean Allemang

The FAMILIAR tool provides disciplined support for collaborative problem solving such as planning or software design. Members of a team use the FAMILIAR visualization modes to examine past cases, construct new solutions, trade off alternatives, and perform "what if" analyses. FAMILIAR includes formal analysis functions that issue advice to help guide the construction and evaluation of solutions.

FAMILIAR allows someone who is facing a complex planning problem to organize the contingencies and alternative plans in a systematic way. It supports reactive re-planning in the face of unforeseen plan failures, by identifying opportunities to re-use plan components (even from plans that were originally rejected). This improves the survivability and fault-tolerance of the plans.

FAMILIAR models problems and their solutions in terms of goals, alternatives, features, and components. Goals represent the objectives of the current task. The alternatives hierarchy is a categorization of known solutions to known problem types. It is a corporate memory of best practice, which may be consulted and borrowed from when confronting new problems. Features are the dimensions along which alternatives differ from each other. Because any complete solution design is a combination of many aspects, the feature hierarchy provides a way to compare and contrast specific aspects of alternative solutions, and to perform "what if" analyses by changing some aspects while keeping others constant.

A solution may be composed of several components, reflecting a decomposition of the problem into smaller, simpler ones (divide and conquer). Of course, choices made concerning the solution to a sub-problem will have an impact on the overall solution. For example, the selection of a particular approach to solving part of the problem may invalidate using certain approaches for other parts. FAMILIAR maintains these dependencies, propagating decisions so that the overall solution remains consistent, and informing the user about the implications of choices made. These implications go beyond a simple yes-no answer as to whether the solution will work. They include tradeoffs indicating how well different goals are met using different solution approaches, and advice on how to fix problems that FAMILIAR has discovered.

FAMILIAR allows a problem-solving team to keep track of several candidate solutions. As alternative solutions are composed, FAMILIAR tracks the rationale for the parts of each solution. It allows team members to evaluate the fault-tolerance of each solution, and to make use of alternative components (either new components or components of other candidate solutions) to address possible failures in the current solution design. In particular, FAMILIAR supports evolution by identifying those parts of the solution that are affected by a change in specifications or operating environment.

FAMILIAR may be applied to a wide range of problem-solving situations. It is particularly useful in situations that are both multi- dimensional (many aspects to the problem) and multi-level (must be decomposed into sub-problems). It is the only decision support technology that manages the interactions between these two sources of complexity on behalf of the decision-maker. As such, it is particularly useful for software design and for planning situations in which there is uncertain, incomplete, or rapidly changing information.

Incremental Constraint Engine: http://www.htc.honeywell.com/projects/dssa/
Honeywell Technology Center – Steve Vestal

The Incremental Constraint Engine provides efficient incremental entry of certain classes of constraints, together with a rapid assessment of feasibility and (in the case of unfeasibility) identification of culprit constraint sets. We have a prototype integration of this capability with DoME to support management of constraints that span multiple models of a system.

Specific demonstrations may include:

- Army AMCOM has created a generic or reference software architecture for the missile domain. This architecture has been captured in MetaH, and the MetaH toolset has been used to analyze the schedulability of the system and produce real-time executables for a variety of target hardware configurations.
- With assistance from Boeing and the Comanche PO, we developed a preliminary MetaH specification for the Comanche Mission Equipment Package avionics. This specification was used to develop a system schedule and perform schedulability analysis.

Related demonstrations may include:

- CMU/Lockheed-Martin have captured a version of their Simplex architecture in MetaH.
- SEI has developed a translator between MetaH and ACME.
- University of Colorado is working to apply their impact analysis to architectures specified in MetaH.

Honeywell has developed a model of portions of the MetaH real-time executive using the MCC/U Mass FLAVERS formal verification toolset.

INSERT - Incremental Software Evolution for Real-Time Systems:
http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/Groups/real-time/insert/
Carnegie Mellon University, Software Engineering Institute and Lockheed Martin Tactical Aircraft Systems - John Lehoczky

The INSERT technology package permits the easy and reliable insertion of new or upgraded capabilities into mission critical systems. This package creates a structured environment based around innovative uses of advanced technologies including analytic redundancy, dynamic component binding, dependency tracking, and data fusion integrity processes. The INSERT approach leads to reduced development and test time, reduced cost, and a reduction of the technical risks involved with system evolution.

The EDCS-INSERT demonstration highlights the application of these tools and capabilities within the context of an F-16 mission system suite. We will be presenting a videotape of a capability upgrade to the F-16 mission system. The demonstration takes place in a ground-based simulation of the actual F-16.

The upgrade involves the insertion of an automated Air-to-Ground weapon delivery capability. During execution of the new capability, a sequence of different failure occurrences will be injected; however, the INSERT architecture will be able to cope with these failures and maintain stable system operation. This capability promises to increase the effectiveness and survivability of the aircraft.

The INSERT demonstration highlights the following:

- The INSERT approach is effective on DoD scale problems. The design properties are relevant and resource efficient.
- The architecture is cost effective. Adoption costs are small.
- The work has resulted in additional architectural approaches that are applicable to a wide range of mission system domains.

The On-Site Demonstrations of Major Components of the INSERT Package include:

- Reliable Upgrade Environments: INSERT F-16 Autoguidance Applications

In addition to the videotape presentation, we are presenting live demonstrations of INSERT capabilities in pure virtual simulation environments. Two demonstration environments will be shown based on an existing USAF/Lockheed Martin F-16 simulator. One environment demonstrates an autopilot upgrade environment designed and generated using Honeywell MetaH on Windows NT 4.0. The use of MetaH (an EDCS technology) supports cross-platform environments and additional architectural analysis. The other environment uses custom code on a Real Time Operating System to provide similar capabilities. Pilot-Relief and automated landing modes are demonstrated.

- Computer-aided Support Tools for Verification of INSERT Switching Rules

INSERT switching rules provide protection against semantic faults that could be introduced in the software upgrade process. The performance of the switching rules over the entire range of possible operating states can be verified using a new tool for modeling and computer-aided verification of hybrid dynamic systems. The capabilities of this tool will be illustrated for the INSERT F-16 autopilot demonstration system. The demonstration includes the building of the model of the hybrid system dynamics, the evaluation of the results of verification queries, and the use of automata approximations to the hybrid system dynamics.

- Semantic Dependency Analysis Tool

Application errors due to hidden side effects are addressed through design-time analysis of a system model. By documenting assumptions about semantic and time-sensitive characteristics of components, the system model allows identification of violated assumptions, inconsistent INSERT configurations, and the impact of the proposed changes. The analysis tool is implemented on ACME/ARMANI, an EDCS technology.

Internet-Based Information Management Technology:
http://www.psl.cs.columbia.edu/current.html
Columbia University, Dept. of Computer Science - Prof. Gail E. Kaiser

Columbia University will demonstrate its Internet-based information management technology for multi-organization collaborative work. Applications are not limited to very large systems engineering, e.g., open-source software and multiple (sub) contractor projects, but also include decision support and distance learning. The main components and toolkits include: Worklets, a mobile agents approach to meta-workflow for dynamic reconfiguration and knowledge propagation; Workgroup Cache, for zero-latency knowledge propagation among dynamically organized groups according to task-specific criteria; Xanth, an XML-based data fusion service; Groupspace Controller, an object/event broker featuring vetoable events and wraparound service activation; TreatyMaker, a toolkit for rapidly constructing and dynamically managing N-ary interoperable alliances among peer services and systems;

JPernLite, transaction management middleware supporting plugin extended transaction models, e.g., for groupwork and "what if" transactions; and TaskWeb, an open hypermedia system for PDA's.

These and other technologies are integrated in CHIME, Columbia Hypermedia IMersion Environment, a framework for generating and managing MUD-like 3D virtual worlds for collaborative information understanding and interaction. Demo scenarios will include software development and multi-agency emergency response.

Jakarta Tool Suite (JTS):
http://www.cs.utexas.edu/users/schwartz/proj.htm
University of Texas at Austin - Don Batory

The Jakarta Tool Suite (JTS) is a set of Java-based tools for developing product-line architectures, application generators, and compilers for domain-specific languages.

JTS is being used in to develop the next generation of FSATS (Fire Support Automated Test System) that has been developed by the University of Texas Applied Research Laboratories. Unlike its predecessor, the new system, FSATS99, is highly extensible: a product-line of FSATS99 simulators can be assembled from components using JTS. Building C2 simulators in this manner appears to have significantly simplified software development, understanding, maintenance, and evolvability of the system. We will demonstrate our current prototype of FSATS99, along with other capabilities of JTS.

Knowledge Depot: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

Knowledge Depot supports Project Awareness by capturing documents and communications relevant to a project and, based on the interests people register with the system, redistributes summaries of this information to interested people. This approach allows a user to specify what kinds of information affects his or her work and skim through summaries of this information to determine if any of the documents or communications have a potential impact upon their work. A user can not only be alerted when there is a change in a component (or other aspect of a project) that he or she depends upon, but can become aware of the fact that the change is being discussed. Project awareness enables the person to prepare for change, and to contribute to the discussion from the perspective of how the change will affect that individual's work.

Little-JIL: http://laser.cs.umass.edu/perptest/
University of Massachusetts at Amherst - Leon Osterweil/Lori A. Clarke

We will demonstrate Little-JIL, a graphical language for specifying the execution of complex processes by teams of agents that consist of both humans and machines. Little-JIL can be thought of as a multi-agent coordination system, or as a process execution system, that offers substantial enhancements over typical workflow systems, especially in its powerful facilities for handling exceptions, its comprehensive treatment of resources, and the uniform way in which it treats humans and automated agents. Little-JIL process specifications are hierarchical decompositions of steps, where steps are guarded by prerequisites and postrequisites, whose violates throw exceptions. Steps also incorporate data flow specifications, as well as specifications of resource types that are used as the basis for dynamic scheduling of resources.

The demonstration will show the use of the Visual-JIL graphical editor for developing Little-JIL process descriptions, as well as the Juliette environment for supporting execution of Little-JIL processes on a distributed platform of workstations. As part of the demonstration of Juliette we will focus on our resource management system and our Grapevine agenda management generation system. Example Little-JIL processes to be demonstrated will include processes for multi-user software design, multiagent negotiation, and perpetual testing.

Maude:
http://www-formal.stanford.edu/clt/ArpaActive/summary.html
SRI/Stanford University - Jose Meseguer/Carolyn Talcott

SRI/Stanford University will demonstrate formal interoperability of ADLs using the Maude tool. Maude is a high-performance reflective language and system supporting both equational and rewriting logic specification and programming for a wide range of applications. Maude can be used to build executable formal models of system architectures very quickly, at diffferent levels of abstraction and amenable to a wide range of static and runtime analyses including model checking, symbolic simulation, monitoring, and theorem proving.

Model-Based (Systems) Architecting and Software Engineering (MBASE): http://sunset.usc.edu/
University of Southern California Center for Software Engineering (USC/CSE), The Aerospace Corp., and TRW - Barry Boehm/Neno Medvidovic

MBASE is a set of guidelines that describe software engineering techniques for the creation and integration of development models for a software project. The models to be integrated extend beyond Product (development) models such as object oriented analysis and design models and traditional requirements models, to include Process models such as lifecycle and risk models, Property models such as cost and schedule, and most notably Success models such as business-case analysis and stakeholder win-win. The approach used in MBASE ensures that a project's success, product, process and property models are consistent and well integrated. MBASE core model frameworks guide the project's convergence on a consistent and feasible set of models, and guide the product's development or enhancement through an extension to the original Spiral Model. MBASE is highly compatible with Rational's Unified Software Development Process, which has adopted the MBASE anchor point milestones (MBASE has adopted Rational's Inception/Elaboration/Construction/Transition phase definitions for the activities between the milestones) MBASE is trying to extend Rational-USDP'S architecture-centric use case-driven process toward a process which is both architecture-and stakeholder-centric, and both use-case-and business case-driven. MBASE provides a constructive approach and extensible framework of collaborative tools, enabling a system's stakeholders to rapidly develop mutually satisfactory (win-win software system solutions.

The tool framework includes the USC Center for Software Engineering's WinWin, COCOMO II, Architecture Attribute Analysis Aid (A4), and Distributed Collaboration and Prioritization Tool (DCPT). It has been integrated with such other DARPA tools as Rapide (Stanford), C2 (UCI), ScenIC (Ga.Tech), MediaDoc (USC-ISI), JWatch (Intermetrics), and Catalyst (MO); an Aerospace Corp. trajectory simulation and visualization program; and with such commercial tools as Rational Rose, CUSeeMe, and RealPlayer.

MBASE and its tools have been applied on over 50 digital library projects at USC. Early adopters of MBASE capabilities include the Air Force C2ISR Center, FAA, The Aerospace Corp., TRW, Litton, and Xerox. The demonstration will show how these capabilities can be applied to a DoD quick response mission requiring not only rapid mobilization and deployment, but also rapid software change coordination and implementation.

MediaDoc: Automated Generation of Multimedia Explanatory Presentations
http://www.isi.edu/isd/I-DOC/media-doc.html
University of Southern California/Information Sciences Institute (USC/ISI) - Lewis Johnson & Stacy Marsella

Many organizations face a constant problem of obtaining accurate, relevant information about complex, evolving systems. We see this with deployed hardware and software systems, for example. Such systems often have long lifetimes, while staff turns over frequently. New staff assigned to tasks such as maintenance and upgrades have difficulty obtaining the information that they need to perform their specific tasks. They waste time sifting through irrelevant information in voluminous documents, and the information that they find may be out of date. Tactical decision-makers in the military face similar problems, but on compressed time scales. Command staff members need to obtain focused views of an involving tactical situation, and their jobs can be hampered if the information that they receive is cluttered with irrelevant material or is out of date.

The MediaDoc project addresses these problems through the automated generation of focused multimedia presentations of multidimensional information.

A user performing a particular task can pose queries about a system; MediaDoc extracts information relevant to the query and the task, and automatically generates presentations combining text and graphics. Automated text extraction tools are also provided that extract semantic information from relevant textual documents, so that this information can be integrated into the presentations. We will demonstrate how these techniques may be applied to a software engineering task and to crisis management in a military relief operation.

MetaH: http://www.htc.honeywell.com/metah.
Honeywell Technology Center – Steve Vestal

MetaH is an architecture description language and toolset for avionics and other real-time applications. The toolset includes a software/hardware binder, schedulability modeling and analysis, reliability modeling and analysis, and partition impact modeling and analysis. There is also an automated composition tool that builds a real-time executable from an architectural specification and a set of software components.

Model Integrated Computing (MIC):
http://www.isis.vanderbilt.edu/
Vanderbilt University/Institute for Software-Integrated Systems - Gabor Karsai

The MIC Environment developed at Vanderbilt/ISIS will be demonstrated through an application: the Integrated Test Information System (ITIS). IT IS has been developed for and is being used by Arnold Engineering Development Center (AEDC), Arnold AFB in support of engine and airframe ground testing. The ITIS integrates many facility-wide legacy data systems and databases, facilitates user-defined analysis of the information, and delivers resulting information to a geographically distributed set of end-users.

The MIC technology has been used on two levels:

- On the "meta" level, the concepts and semantics of the application domain, and their mapping into the implementation domain have been captured using the meta-level modeling environment (which is just another instance of the MIC architecture). This was followed by the synthesis of a domain-specific modeling and program generation environment, and the development of the run-time support system for the application.
- On the "domain" level, end-users use the environment to develop domain models that are then used in the synthesis of the actual test-specific ISIS application.

The DARPA EDCS technology provides the support for meta-level modeling of the domain, the generation/synthesis of the domain-specific environment, the infrastructure for the domain-specific environment, and the generation technology for synthesizing the application from domain models. The demonstration also integrates other EDCS technologies. For instance, ACME architecture models are generated, which then can be analyzed using ACME tools.

The demonstration shows the meta-modeling environment, the domain-modeling environment, and the generated application. The demonstration also shows how a small-scale, but fully functional ITIS system can be customized on the fly using the meta-level technology.

To show cooperation with other DARPA programs and the dissemination of the technology, another demonstration shows how a domain-specific modeling and analysis environment can be built (and used), for Adaptive Computing Systems (ACSs). In ACSs, dynamically configurable (and re-configurable) hardware and software architectures are modeled, design alternatives analyzed and are explored, and application hardware and software are synthesized.

ORBIT/VIRTUE - Collaboration and Visualization Support for Complex Systems Evolution:
http://www.dstc.edu.au/wOrlds/
University of Illinois/University of Queensland – Daniel Reed/Simon Kaplan

This demonstration illustrates the integration of desktop and virtual environment collaboration systems and their application to complex problems, such as software engineering, logistics and data analysis. The Habanero and Orbit desktop systems support extensible, varying intensity collaboration and tool sharing across local and wide area networks and are coupled via shared controls and streaming audio/video to the Virtue virtual environment for

high-modality data visualization, analysis, and manipulation. In addition, mobile, handheld devices allow collaborators to participate as equals with those at desktop and virtual environment stations. The complete environment provides rich, varying-modality support for complex collaborative work situations.

This year's demonstration will also demonstrate integration with various planning and rationale capture toolsets to support complex command and control activities such as disaster relief.

High Assurance Technologies: http://www.cs.uoregon.edu/~michal
University of Oregon, Dept. of Computer Science - Michal Young

The value of verifying high-level architectural designs is magnified if one can also demonstrate consistency of the verified architectural model with the actual implementation, and do so without imposing unreasonable constraints on designers. University of Oregon will demonstrate an approach for verifying and maintaining potentially complex relations between an architectural design model and running code. Demonstrated capabilities will include aids to recovering architectural design information, restructuring "as-built" system organization into a logical structure more suitable for static verification, and supporting conformance testing of implementations using test oracles derived directly from verified architectural design models. Support for dynamic testing extends beyond product delivery with low-impact "residual" monitoring of test obligations.

Software Architecture, Analysis, Generation, and Evolution (SAAGE): http://sunset.usc.edu/
University of Southern California Center for Software Engineering (USC/CSE), The Aerospace Corp., and TRW - Barry Boehm/Neno Medvidovic

USC/CSE's Software Architecture, Analysis, Generation, and Evolution (SAAGE) project focuses on consistent transfer of architectural decisions into designs and implementations and architecture-based evolution of large-scale systems. Major goals of this work are:

- evolution flexibility and precision,
- simplicity and formality of architectural descriptions,
- architect's discretion in interpreting analysis results, and
- self-evolvable tool support.

To this end, USC/CSE provides an integrated toolset, composed of in-house and third-party tools (DRADEL and RationalRose, respectively). The integrated toolset enables:

- style-based application design and implementation,
- component-based architectural composition,
- architecture modeling and analysis,
- architecture-based software evolution,
- consistent refinement of architecture into design, and
- system generation.

Quest: http://www.mcc.com/projects/quest
Microelectronics and Computer Technology Corporation (MCC) – Tim Harrison/Debra Richardson

MCC will be demonstrating version 4.0 of the Quest Toolset consisting of a set of software analysis and testing tools integrated within a common user interface. The tools allow the developer and/or test engineer to statically analyze source code to discover dependencies between program statements, modules, or systems, and statically or dynamically verify the existence or absence of specified program behaviors. Code dependence analysis is of particular value for code understanding, legacy system analysis, and selective regression testing. Program behavior verification, static or dynamic, is of extreme value for quality, reliability, and conformance assurance. The toolset is able to analyze both Ada and C/C++ source code and programs. Capabilities beyond that of version 3.0, demonstrated last year, are inter-procedural analysis, improved performance, and improved usability of the toolset.

Securely Wrapping COTS Products: http://www.isi.edu/software-sciences/multi-gen/multi-gen.html
University of Southern California Information Sciences Institute (USC/ISI) - Bob Balzer

We will demonstrate a WindowsNT based wrapper technology that allows Commercial Off-The-Shelf (COTS) products to be integrated with each other, extended to utilize externally supplied capabilities, and restricted to operate within user defined security policies. The demonstration will include incorporating encryption, a virtual file system, and resource limitations into a variety of COTS products. It will also demonstrate safe execution environments for safely using active content in web browsers and office products.

Siddhartha - Automated Test Driver-Oracle Synthesis: http://www.ics.uci.edu/~djr/edcs/PerpTest.html
University of California, Irvine - Debra Richardson

Software design decisions must balance a collection of functional and non-functional software quality attributes (e.g., correctness, efficiency, testability, and analyzability). The tacit assumption made by purveyors of general-purpose software test tools is that testability dominates software design decisions, which is unreasonable for many application domains. This mindset engenders the notion of software design for testability (DFT) and often requires that the interfaces of units under test (UUTs) be fully parameterized to support complete control and observation of behavior via programmatic invocations of the UUT. While such a program design style supports testability, it is eschewed in several important application domains (e.g., digital avionics and flight controls) because it reduces run-time performance and the ability to estimate run-time performance a priori.

This problem is usually solved by iterative, ad hoc, development of domain-specific test development tools and languages. Such tools and languages are "home-grown" within software development organizations and support test development in a manner that respects business-prioritized software quality attributes. Unfortunately, little guidance for developing such technology can be found in the software engineering body of knowledge. Siddhartha is a defined, disciplined, alternative technique for developing domain-specific test tools that fit within a software development

organization's business context and respect legacy languages, design styles, and development processes. Siddhartha develops domain-specific test automation support for transforming formal test specifications (TestSpecs) into test driver-oracle procedures (TDOPs), where a TDOP invokes not only the unit-under-test (UUT) but also an embedded oracle that verifies whether the tested UUT behavior agrees with the expected behavior expressed in TestSpec. In using Siddhartha, a test engineer develops a domain-specific synthesizer to accept and generate formal test artifacts in formats, languages, and styles already in use in her application domain. Thus, Siddhartha provides test development automation support in application domains that cannot be well-supported by general-purpose test development tools.

Siddhartha has been applied to produce two domain-specific test synthesizers to date: Siddhartha-SCR(SCR*log, Ada) and Siddhartha-regression(Ada, Ada). Siddhartha has been validated against a significant, real-world example: the Ada operational flight program (OFP) for research flight control system (RFCS) of the production support flight control computer (PSFCC) used on the F/A-18B Hornet system research aircraft (SRA) operated by NASA Dryden Flight Research Center (DFRC).

SoBeIt: Structural and Behavioral Execution Instrumentation Tool: http://www.ics.uci.edu/~djr/edcs/PerpTest.html
University of California, Irvine - Debra Richardson

SoBeIt supports development and assurance of quality software through automated specification-based testing. SoBeIt supports software testers with the following capabilities:

- persistent test artifact development and maintenance;
- fully automated testing, including monitored test execution; formal, automated test result checking via specification-based test oracles;
- functional and structural test adequacy definition and measurement.

The tester provides SoBeIt with the component-under-test (CUT) and corresponding component specification (CS) along with a persistent test suite (TS) and SOBEIT does all the rest: the Instrumentor takes a representation mapping relating events in CS to occurrences in CUT and instruments the CUT to collect event traces; the instrumented CUT is automatically executed on all test cases specified in TS; the test executions are monitored to collect the required event traces; an FSA representation of the specification CS is interpreted on the event traces, thus serving to compare the test execution results to the specification; pass/fail results are reported.

SoBeIt is written in Tcl/Tk, although it integrates components written in a variety of languages. The Instrumentor currently works for Ada83, although instrumentation can be done manually. The OracleGenerator currently works for GIL, but FSAs can be provided directly via an FSA Editor. The rest of SoBeIt is language independent. SoBeIt is a scaled-down version of ROSATEA's TAOS environment serving to prototype structural and behavioral instrumentation capabilities.

Software Composition Workbench: http://www.csee.wvu.edu/~resolve/scw/
West Virginia University- Murali Sitaraman/Steven Atkinson

The central objective of the software composition workbench is to enable reliable component-based software engineering. The workbench provides a framework for construction of systems based on reusable components, and formal and modular demonstration of correctness. Typical components are parameterized, object-based, and are characterized by formal behavioral specifications and alternative performance-flexible implementations. The workbench enforces a rigorous discipline for component and subsystem construction. Salient features of the workbench include:

- A conceptual model of software construction
- Use of wizards for valid and syntax-free composition of components
- Formal and modular reasoning of component correctness using mathematical specifications, and theorem proving
- Run-time interface violation detection through generated wrapper components
- Code generation in alternative programming languages such as Ada and Java

The workbench is based on the RESOLVE framework, discipline, and notation. For related details and publications, visit our Reusable Software Research Groups (RSRG): RSRG at The Ohio State University and RSRG at West Virginia University.

Specware: http://www.kestrel.edu/HTML/prototypes/specware.html
Kestrel Institute - Jim McDonald

Kestrel will demonstrate a variety of uses of EDCS technology in the Specware specification and design system.

We will describe our technology and explain the potential benefits to real-world applications: correctness, security and dramatically enhanced productivity.

In particular, we will show rapid automated derivations from specifications of several provably correct programs--for example, various schedulers and a Java byte-code verifier. These are substantial programs containing several thousand lines of Common Lisp or C++ code.

We will show how new versions of those programs can be rapidly generated after modifications to the functional specifications. For example, we will modify a scheduling specification to include additional constraints and then quickly generate a new application meeting those constraints.

We will also show how new implementations of each version can be rapidly generated after modifications to the implementation specifications. For example, we will respecify the implementation of some data structure from one format to another, rapidly resynthesize the overall application, and show how performance is affected. We will demonstrate applications developed in Specware by teams at Boeing, Motorola, and elsewhere.

TestTalk: Software Test Description Language: http://www.ics.uci.edu/~djr/edcs/PerpTest.html
University of California, Irvine - Debra Richardson

Software tests are valuable intellectual assets, especially in long-lived, multi-version, multi-platform commercial software. The highly-publicized Y2K software problem provides a very good sense of the problems that arise in such a domain as well as how long software tests should last. Software tests represent significant investment. Test developers are generally on their own to determine how to write better automated software tests. This leads to a number of problems, including: (1) Understandability: test cases and test oracles are typically buried in test code and hence difficult to rediscover; (2) Maintainability: automated tests are extremely sensitive to changes in the implementation. Both problems lead to difficulty in adjusting a legacy automated test because of the arbitrary nature of the current practice of test code development.

TestTalk is a software test description language designed for specifying test cases and test oracles in a manner natural to the software testing process rather than the programming or development process. TestTalk helps testers focus on requirements and design aspects of software tests rather than the implementation details of test execution. By enabling practitioners to separate concerns between software test description and test execution, TestTalk provides the means for creating software tests that are readable, maintainable, and portable, yet executable.

The ultimate goal for TestTalk is to support the following maxim: "Write Once, Test by Anyone, Anytime, Anywhere, with Anything". By "write once", we mean that test descriptions only have to be written once but can be used perpetually. New transformation rules evolve old tests to account for various changes. By "test by anyone", we mean that TestTalk descriptions are so easy to understand that a tester can easily take over tests written by other testers or developers. By "anytime", we mean that TestTalk tests survive over time through application evolution and revisions. By "anywhere", we mean that TestTalk tests can be transported to another platform or operating system without modification. By "with anything", we mean that switching the test automation environment does not nullify TestTalk tests.

We are building a toolset to support the TestTalk language. The current toolset consists of a prototype parser and translator, which recognizes test descriptions and transformation rules (both expressed in what we consider the core language). The TestTalk toolset produces automated test programs for the application-under-test for a specific platform and test automation tool by using the transformation rules in the translation of the test descriptions.

UML/Analyzer - A System for Defining and Analyzing the Conceptual Integrity of UML Models: http://sunset.usc.edu/
University of Southern California, Center for Software Engineering (USC/CSE) – Alexander Egyed

Software development is about modeling a real problem, solving the model problem, and interpreting the model solution in the real world. In doing so, a major emphasis is placed on mismatch identification and reconciliation within and among system views (such as diagrams). UML/Analyzer describes and identifies causes of architectural and design mismatches across UML views as well as outside views represented in UML (e.g., C2 style architectures).

- It is integrated with Rational Rose (market leader for OO modeling)
- It implements a generic view integration framework
- It incorporates UML's Object Constraint Language (OCL)

UML/Analyzer supports the definition of mismatch rules and model constraints. It also defines what information can be exchanged and how it can be exchanged. With that, architects can identify and resolve inconsistencies between views automatically:

- Mapping: Identifies related pieces of information and thereby describes what information is overlapping and can be exchanged.
- Transformation: Extracts and converts model elements of views in such a manner that they can be interpreted and used by other views (how to exchange information).
- Differentiation: Traverses the model to identify (potential) mismatches within its elements. Mismatch identification rules can frequently be complemented by mismatch resolution rules.

UML/Analyzer is integrated with Rational Rose and is used to create and modify views (synthesis). Rational Rose models are converted through an automated process into UML-A where they are analyzed via UML/Analyzer. Model constraints and mismatch rules are verified via a parser component. The conceptual integrity of the model is then validated through the model checker component. The model checker makes use of mapping, transformation, and differentiation. Generated modeling information as well as identified model mismatches can be fed back into Rational Rose for visualization.

- UML/Analyzer identifies inconsistencies and incompletenesses,
- Model currently supports class, object, sequence, collaboration, state, and various architectural diagrams (e.g., layered and C2)
- Model constraints, mismatch rules, and transformation rules can be modified without programming.

WebDAV: http://www.ics.uci.edu/pub/edcs/
University of California, Irvine – Richard Taylor/David Redmiles

WebDAV is an extension of HTTP that provides a standard infrastructure for asynchronous collaborative authoring of a wide variety of content across the Internet. WebDAV has been approved by the IETF and is being actively developed by a number of Software vendors, including Microsoft, IBM, Xerox, Novell, DataChannel, and CyberTeams. This demo will feature a WebDAV client (WebDAV Explorer) which will show how the WebDAV protocol facilitates collaborative use of distributed files.