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).

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

SoBelt supports development and assurance of quality software through automated specification-based testing. SoBelt 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 SoBelt 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.

SoBelt 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 SoBelt is language independent. SoBelt 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.