Exhibits

The following exhibits are anticipated

E1. Nu Thena Systems, Inc.

Mike Vera <mike@nuthena.com>
Peter Stoupas <peter@nuthena.com>

Nu Thena Systems, Inc., plans to exhibit the latest release of Foresight, Version 4.00, which incorporates several additional tools, capabilities and features, including a user-configurable user interface, enhanced data visualization and documentation capability. Visitors to the booth will be able to see a live product demonstration of Foresight Version 4.00, which includes Visual Numerics Inc.’s product, PV-WAVE and Altia’s popular design tool set. The data visualization tool has been incorporated for plotting data captured from Foresight model simulations. Besides the standard Foresight capabilities, the user can plot various graphs and charts from the data collected using Foresight’s monitor facility, including timelines, histograms, strip charts, xy plots and diagrams. Altia’s design tool set provides a human interface prototype tool. Foresight users can now rapidly create a graphical prototype of a product’s human interface and dynamically link it to the Foresight system models. Pre-defined graphical, animated models (such as meters, dials, menus, etc.) may be linked to the user’s Foresight system models, allowing users to interact through the graphical interface. Additionally, Nu Thena will exhibit new automated documentation facilities and capabilities that allow users to document an entire system or subsystem with one command.

Output is automatically generated in either encapsulated postscript or marker interchange format for easy integration with customer publishing tools. Foresight’s data collection and monitoring capabilities have been improved in Version 4.00 to facilitate model analysis and simulation. Foresight Version 4.00 includes new modeling constructs, which allow the user to more easily capture and analyze complex system behavior at all levels of abstraction. The Nu Thena customer base has increased by 50% during the last two years, and continues to grow at a rapid pace into application areas such as telecommunications, avionics, aerospace and defense, medical imaging, transportation and business systems, and more. Nu Thena remains at the forefront of system design and simulation, and is committed to providing engineers with the most comprehensive and adaptable design and simulation tool available today. The company’s international headquarters are located in McLean, Virginia, with distributors throughout Europe and Australia. In addition to a range of system design products, Nu Thena provides training, consulting and product maintenance services to ensure that automation tools are rapidly assimilated into a customer’s organization and system development process.

E2. Engineering Baselines in System Development

George Mason University Fairfax, VA
Richard P. Evans <revans@gmu.edu>
Soojong Park <spark@gmu.edu>
Mark Merriman <mmerriman@gmu.edu>

Engineering Baselines

Engineering baselines represent a new and needed paradigm for controlled visibility and traceability in system development. One of the primary features of this paradigm is the enabling, in standard ASCII files, of the participation in, and thus the use by, everyone involved at all levels of the development. Engineering baselines enable and support significant improvements in system development [and also software and other system component reuse] by three primary features: (1) establishment of all system descriptions in plain ASCII files; (2) application of three types of numbers: a unique system number for all system elements, an engineering tag to establish links among system elements, and change set numbers; (3) use of plain ASCII two-column index files for all paired links among system elements.
Engineering baselines, in contrast to “official” contract baselines, may be based on a wide variety of boundary conditions, change sets, etc. The availability of controlled “engineering” baselines for timely in-parallel engineering, including assessments of alternatives as well as initial allocations, system planning, drafts, etc., overcomes the constraint of being limited to official contract baselines as the only “controlled” set of baselines.

Effectiveness and efficiency in system development requires machine processing of many forms of system descriptions. Integration among system elements, in turn, requires a common baseline of element descriptions. Engineering baselines include the structuring, with a common baseline numbering, of system descriptions into plain machine processable ASCII files.

**ASCII Files – System Descriptions**

Engineering baselines establish standard structured ASCII representations of documentation—not bulleted and otherwise formatted. Change identifiers are also linked to each system element—not, for example, just change pages—with an identifier on the page but not for each element. That structuring includes converting tabular data to whole statements and concatenating lead in information to bullets to make them independent standalone entities. Each system element represents one statement or item, and can be understood without direct reference to its context. System elements are aggregated or decomposed as warranted. For example compound sentences may be decomposed into separate simple sentences, and sets of so-called “shall” statements may be aggregated into one system element.

**Engineering Baseline Numbers**

Three types of numbers are used in engineering baselines. The most important is the system number, a unique identifier for every system element. The system elements addressed span the complete system, including specifications, designs, hardware components, work breakdown structures, user manuals, development milestones, installation procedures, test cases and scenarios, and software code modules. These numbers establish machine-auditable allocations and links between elements, and record changes to them individually as well as their associations.

The engineering tag permits the definition of links among system elements, including the allocation of system elements to many types of categories, including elements of the Work Breakdown Structure (WBS). The change set numbers establish for each system element links to both the global version from which the element comes, and to each Request for Change (RFC) or Engineering Change Proposal (ECP) that affects this element. The formats and detailed meanings of these numbers are described below.

**Two-Column ASCII-based Index Files**

The two-column, ASCII-based, engineering tag index files, created by any and all engineers using plain word processing packages, enable, by their practicality in day-to-day use by everyone involved, controlled visibility and traceability into the complete development effort. They, in combination with the other features of the engineering baseline paradigm, are one of the keys of its effectiveness.

The key is to facilitate, and thus achieve, on-the-spot recording of links by engineers as they create new system elements. That is information that typically only such engineers possesses, and it is even perishable in relatively short time intervals with those who thus are “on the scene.” Examples of such “linkage information” includes the associations between code modules and design specifications, as well as that between top level and lower level specifications. The engineers “in the field” so to speak, generally do not have specialty database type tools up on their screen, nor do they know how to use them—and DBA privileges must be restricted anyway. But they can prepare plain two-column ASCII files, and these files can, in turn, be used either in aggregate with other two column ASCII index files, or loaded into more central database systems.

Index files, prepared as individual two-column ASCII files, are not only amenable to being aggregated into larger sets of other plain ASCII files, they may also very properly be aggregated into centralized specialty database-oriented software packages. While not in any way constraining the use of specialty database-oriented tracing
approaches, the index files actually enable them by enabling wide preparation and use outside of, and thus in support of, central database-oriented systems.

On the other hand, using only specialty software applications, rather than ASCII index files to create as well as maintain links, restricts visibility into those links to either hard copy tables or direct use of the specialty software that created the table. Individual index files, however, remain visible to any and all for use, modification, extension, and review, and on any machine, and simultaneously also provide the needed inputs for a central database repository or report generator, as may be desired.

System Numbers

System numbers are unique for each system element, including specification elements, software code, drawings, and hardware elements. System numbers and associated tags, maintained in separate two-column ASCII-based index files, can be assigned by system developers when they create new system elements, without using specialized tools, or can be assigned using database or CASE tool systems.

When a system element in a document changes, the next available system number within that document is assigned. Any previous system numbers associated with the system element are retained. When all system elements have system numbers, many aspects of system development can be readily tracked, including testing, allocating, auditing, developing, training, and even funding; all activities refer to the same system number. System numbers, as standard sequence numbers for all system elements, also enable machine-based audits of all system elements and their associated links and allocations. A typical 3.4.2.2 numbering system cannot be machine audited: a machine process knows that 6 follows 5 but it does not know that, in a particular case, 2.3 follows 2.2.11.

Engineering Tags and associated Index Files

The engineering tag establishes explicit relationships between system elements. Engineering baselines and associated engineering tag index files are maintained as ASCII files, and thus can be managed with almost any tools or environments. The Engineering tag is of the form [xxxxxy.zzzzz], where the components are: xxxxx is the tag prefix, it may be a test case identifier, the configuration item to which a system element is allocated, the allocation from specification element to design element, or the allocation from a system specification to a lower level segment specification.

y is a character indicating if engineering cognizance is to maintained for the system element. The use of the alphabetic character in the engineering tag is also centrally controlled. This tag feature overcomes the limitation of “shall”-only approaches and is thus especially critical to the enabling of engineering-controlled designation of system elements to be included in engineering efforts, including allocations and links: engineering studies, audits, designs, and tests. The engineering tag designation of system elements for which “engineering cognizance” is to be maintained overcomes a typical limitation that accountability or cognizance is limited to system element descriptions containing the term “shall.” The capability to designate cognizance selectively for any and all system elements can be especially critical in the controlled decomposition and/or aggregation of a variety of system descriptions, including multiple “shall” statements that actually represent a single class or object, and compound zzzzz, the engineering tag suffix, indicates the type of engineering tag prefix. The tag types are differentiated by this “originator and purpose” suffix. Thus, for example, a tag type of “00005” could be used to establish a list of test cases.

Blocks of tag suffixes are assigned by the program office for use by various organizations throughout the program; with an allocated suffix, those to whom a given tag suffix is assigned generate the definition of their own prefixes. A simple but useful system audit can be supported by aggregations of the two-column ASCII files that establish links between individual tags and assigned system numbers. Such audits can include assurance that all designated engineering tags are used somewhere in the system. In addition, a variety of allocations of system elements to test cases, design items, code modules, etc., can be audited in composite as they use engineering tags with different suffixes.

The engineering tag allows designation of allocations, links, and traces of all types, with links defined in ASCII files which can be generated and processed by any type of software application and thus by any engineer. The
creation and maintenance of engineering tags is thus not limited to a constrained set of trained operators using a few licenses for a specialty database oriented software package.

**E3. Telelogic Incorporation**

**SDT**

SDL is an object-oriented, formal language defined by ITU-T (former CCITT) as recommendation Z.100. Development of SDL started in the early. The language is intended for specification of complex event-driven real-time and interactive applications involving many concurrent activities which communicate using discrete signals. The strength of SDL is its ability to describe structure, behaviour and data of a system. Today, SDL is a widely accepted standard. It has from the beginning been used by the industry. Comments from user organisations and the industry have continually improved the language. SDL diagrams are easily understood even by non-technicians. This translates into greatly improved communication between system designer and end user.

SDT toolset covers the entire software development life cycle: requirement, specification, verification, validation, implementation, and testing with ITEX.

SDT is a set of highly integrated software tools for managing and automating the development and maintenance of real-time and interactive systems. SDL (Specification and Description Language) and MSC (Message Sequence Charts) constitute the basis of SDT. It makes it possible to describe the interaction between a system and its environment as well as the system internal behavior.

The strength of SDT has always been the tight coupling of the tools and the high integrity of the generated code, independent of hardware platform. This provides strong support to the designer modifying and testing a complex application. SDT offers comprehensive life-cycle integration resulting in high productivity through accurate design and generation.

Object-oriented facilities in SDL combined with a well-designed user interface, make SDT very attractive. Special consideration has been taken in SDT to minimise the amount of work required to input specifications and operate the tools. This streamlining of the user interface in all tools affects the activity throughout the development life-cycle.

**ITEX**

ITEX supports all the phases in the creation of a TTCN Test Suite. This ranges from generation of TTCN from SDL specifications, editing the TTCN code, verifying, validating and finally executing the TTCN Test Suites. ITEX also supports the maintenance phase in the life cycle of a test suite. The modules and sub-tools in ITEX are closely integrated and are all based on modern user interface techniques. This makes ITEX easy to use and learn as well as being fast and efficient to work with.

Since TTCN Test Suites can be very large, ITEX is designed with high performance in mind. An optimised and efficient representation of the test suite ensures that all operations performed in ITEX are fast. When used with SDT, the SDL Design Tool, the verification and validation phases are enhanced by use of the combined simulation environment. Semi-automatic Test Case generation guarantees correct Test Cases and speeds up the editing process.

ITEX effectively supports the following activities in the conformance testing:

*Development*

In the development phase the Test Suite is created using the editing tools of ITEX. The Export and Import Tools as well as the Compare and Merge Tools within ITEX offer a powerful environment when contents of previous Test Suites are to be reused. The Browser Tool is a powerful navigator for the Test Suite. It presents the relevant parts of the Test Suite according to the user's needs. The Data Dictionary functions of the Table Editor speeds up the editing process and ensures correct syntax.
As an alternative to the TTCN Tabular format, ITEX introduces the Graphical Tree Format, supported in the Graphical Tree Editor. The format is fully compatible with the standard TTCN format and offers an easily intelligible graphical view of Test Cases.

If an SDL specification of the system subject to testing is available, the SDT TTCN Link will provide significant improvements in productivity and quality. The tool allows for semi-automatic Test Case generation based on an SDL specification. It also produces test coverage information viewable in a user friendly graphical format.

**Verification**

In the verification phase the Test Suite created in the previous phase is analysed in terms of the syntax and semantics of TTCN and ASN.1. The ITEX Analyser detects syntax and semantic errors. The complete Test Suite or selected parts can be analysed. Errors found by the Analyser are immediately highlighted directly in the TTCN tables as well as visualised in the ITEX Browser.

In the combined ITEX and SDT Simulator environment the Test Cases can be verified with respect to a system under test defined in SDL.

**Validation**

The joint environment of ITEX and SDT supports a tool set for the validation of a Test Suite. The ITEX Simulator allows for the execution of a TTCN Test Suite in a host environment. The system under test then consists of an SDL system. This system also operates as a Simulator in the host environment. When connecting the ITEX Simulator with the SDT Simulator the combined system is simulated. The Test Cases are executed one by one or in batch. As a result of the execution, test coverage information is produced. This information can be visualised by the SDT Coverage Viewer.

**Execution**

ITEX supports the translation from TTCN to C. This code is automatically generated which means that the number of errors are greatly reduced compared to manual coding. The ITEX C-Code Generator produces code that is independent of both the target system and the application (i.e. protocol). This means that the generated code suits almost any test equipment supporting C. In order to be executable, the generated C-code must be linked together with functions implementing the target system dependent parts and the application dependent parts.

The generated C-code is linked together with the ITEX Simulator library, a user friendly and powerful execution environment is created. The resulting code forms a combined simulator and debugger for TTCN. It allows for single stepping, breakpoint and slow-motion execution.

**E4. Advanced System Technology, Inc**

AST will exhibit its QASE RT system engineering technology at this year's international symposium. The exhibition will include a demonstration and literature will be available for review. QASE RT is a software tool for quantitatively evaluating the performance of a computer system design – hardware, software, communication, data and data placement. Benefits include uncovering performance issues early, thereby avoiding software redesign, hardware upgrades, or both. When a system is deployed. QASE RT can be used to conduct capacity management and planning. QASE RT uses an easy to learn, direct manipulation interface. Initially, a graphical user interface is used to describe the software, hardware, communication media, and data components. The resultant design can then be evaluated using both analytic and simulation models. Finally, the report capability can be used to product presentations and reports. QASE RT is unique in offering both analytic and simulation evaluation from the same system specification. QASE RT's analytic models quickly evaluate a large number of potential solutions. The simulation models provide detailed analysis once the number of candidate solutions have been narrowed down. In either case, you don't have to be a modeling and simulation expert to use QASE RT. QASE RT is priced at $19,500.00 for any of the following platforms: Microsoft Windows NT (Intel); Apple Macintosh and
E5. Virtual Prototypes, Inc.

E6. North Carolina State University

E7. Carnegie Mellon University

E8. Imperial College, UK