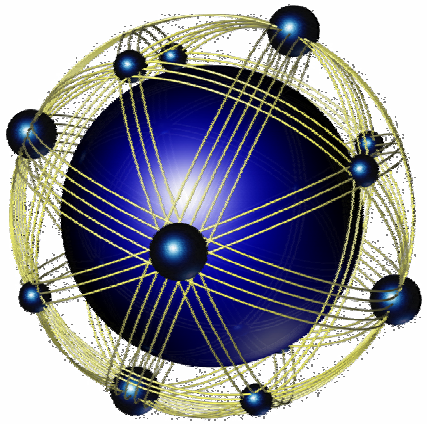
NDIA 8th Annual Systems Engineering Conference

"Automated Software Testing Increases Test Quality and Coverage Resulting in Improved Software Reliability."

October 25, 2005

Frank Salvatore
High Performance Technologies, inc.
3159 Schrader Road
Dover NJ, 07801
(973) 442-6436 ext 249
fsalvatore@hpti.com



Outline

- Introduction
  - Background
  - Project Purpose & Goals
- Overview
  - SW Reliability
  - Statistical Testing
  - Model Based Specification and Testing
- Development Flow
  - Tool Set Architecture
  - Module Review
  - Auto Tester
  - Conventional vs Statistical Testing

Background

- Phase I SBIR Completed in FY 2004 proving feasibility.
- Phase II SBIR to Start in FY 2006
- Sponsor: US ARMY ARDEC, Fire Control Systems & Technology Division (FCSTD)
- Contractors:
  - Cognitive Concepts, LLC Prime
  - High Performance Technologies, Inc (HPTi)
  - Software Silver Bullets

Project Purpose & Goals

- Generate an integrated process which enables any SW Development organization to apply Model based Specification and Testing (MST)
- Significantly advance the state of the practice for system level MST.
  - ✓ Create large models of complex system software behaviors that closely represent expected operational behavior of a specific system.
  - ✓ Automatically generate test cases from the model.
  - ✓ Define and store test scripts associated with every stimulus in the test population.
  - ✓ Generate executable test scripts.
- Implement the required tools that will enable bringing Model Based Specification and Testing technology to market.
- Reduce Software Life Cycle Maintenance Costs.

Overview SW Reliability

- ***Software Reliability*** **- Probability of failure-free software execution in a specified operating environment.**
- ***Software Reliability Engineering*** **- Systems engineering process activities ensuring reliable software systems.**
  - ***Assessment*** **- software reliability can be assessed (measured) only when the software is executing, either in a test lab or in the field.**
  - ***Prediction*** **- prior to having executable software, assessment is done by inference via a forecast.**

SRE Challenges

- Verifying the system does what users want.
- Integrating Requirements analysis and System Software testing.
- Determining what to measure and when to measure it.
- Limiting scope and breath of testing to stay on schedule.

SRE Fundamental Principal

SRE involves:

- **Developing an operational, or usage, profile of the software system under test and**
- **Exercising random test cases from the profile to obtain a direct assessment of the reliability of a software system**

Statistical Testing in a Nutshell

Statistical Testing

- Specification represented in the form of usage models
- System tests generated directly from usage models

Markov-chain usage models

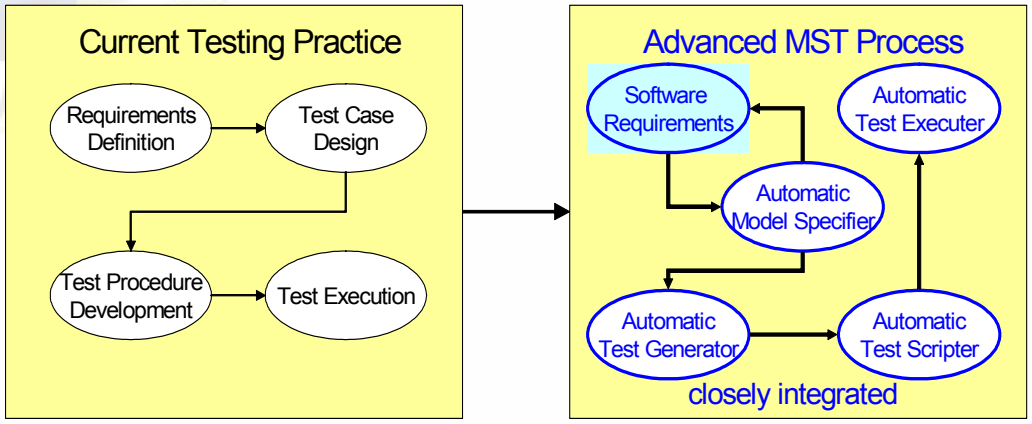
- Black box state-based models that cover every possible state of *usage* for a software system
- External behavioral representation of system
- Composed of states (conditions) and arcs (stimuli)

Software tool generates random test cases

Current State of System Software Testing

Industry practice for testing military applications uses a requirements-based approach.

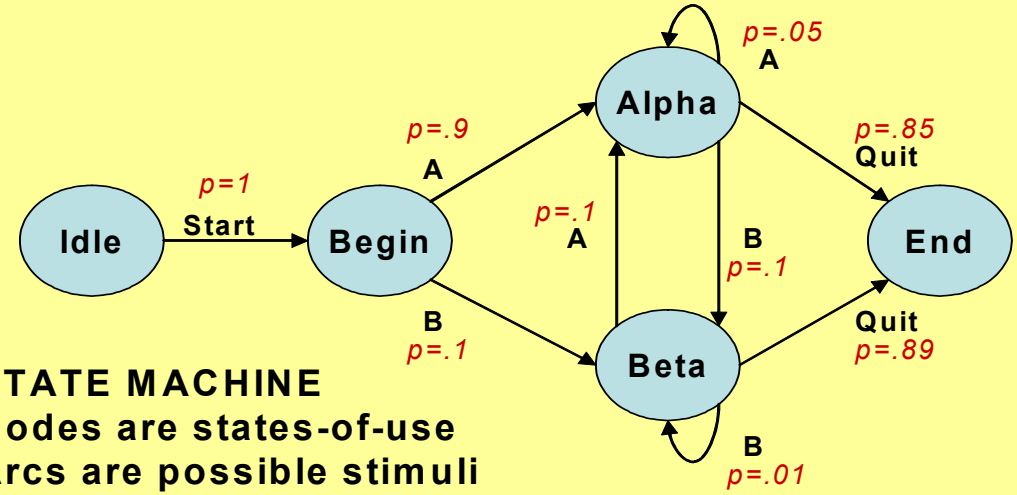
- Test cases are defined for each requirement, or shall statement.
- Test cases are designed manually or with a software tool that is independent of the requirements tool.
- Test cases are scripted manually or with a tool that is not integrated with the test design tool.
- Tests are executed manually or in some cases the tests are automated utilizing a project specific test automation tool.



An innovative approach to requirements specification and testing

MBT Structure

- MBT is a black box representation of the expected behavior of system software.
- A model-based specification is called a usage model specifying how the system is used, or behaves.



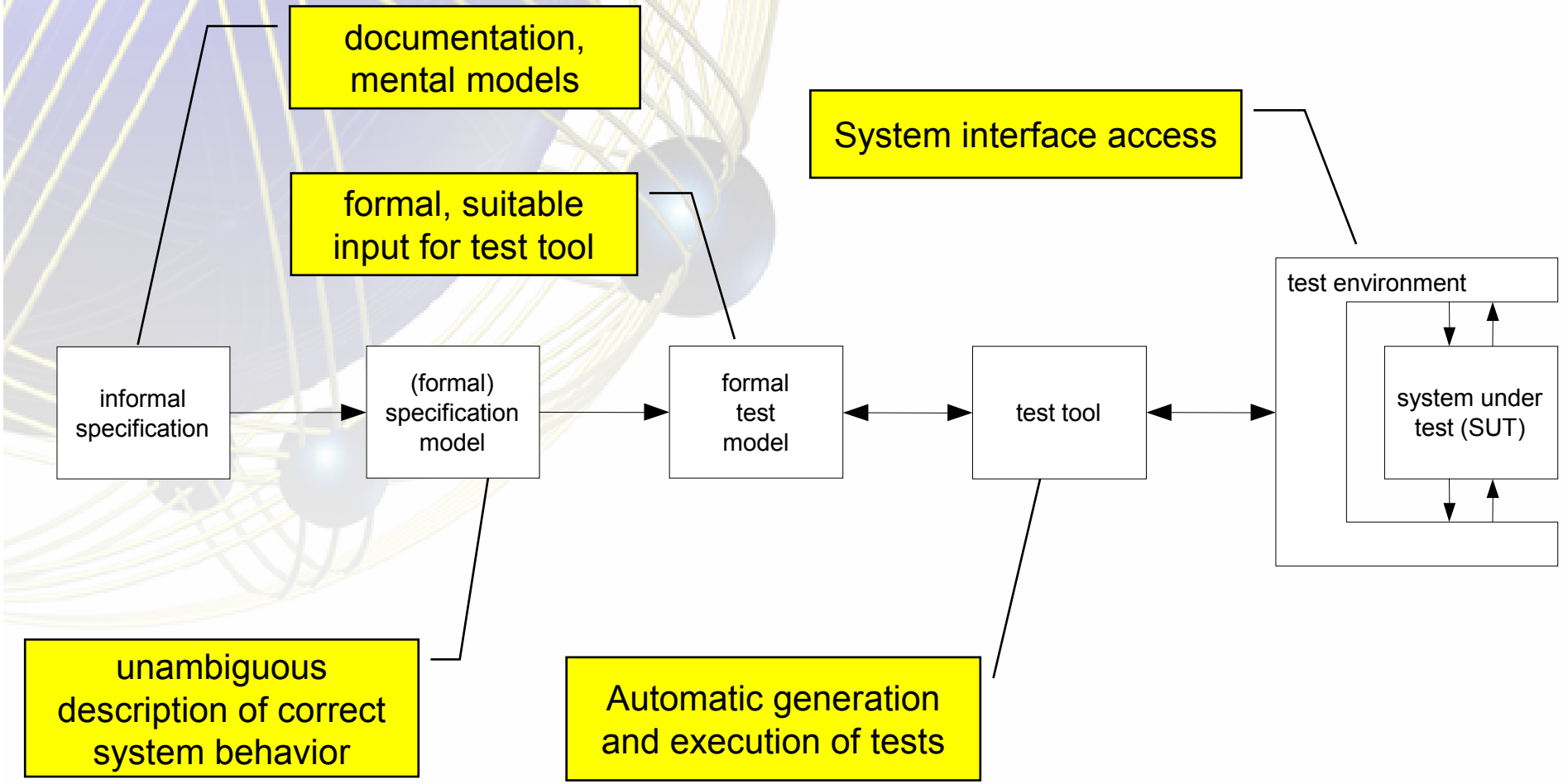
STATE MACHINE
Nodes are states-of-use
Arcs are possible stimuli
Probabilities (*p=1*) define expected usage
Test case is a path from initial to terminal state

MST Overview

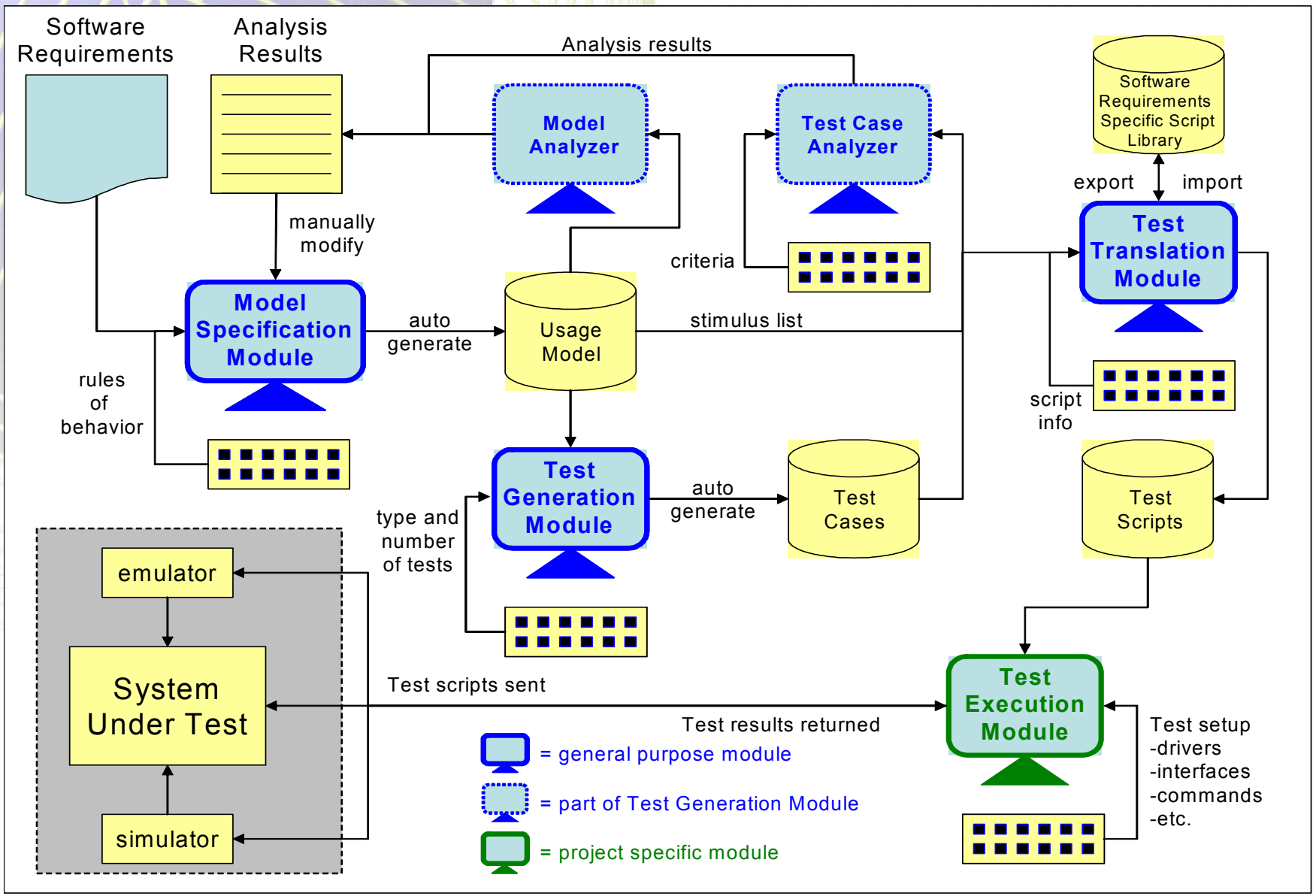
MST

- Provides a structured approach to requirements analysis and software test design.
- Ensures the system specification prescriptive and consistent to enable automatic generation of system software test cases.
- Facilitates an objective assessment of system software reliability.
- Enhanced communication between developers and testers.
- Eases the updating of test suites for changed requirements.
- Shorter schedules, lower cost, and better quality.
- A model of user behavior.
- Early exposure of ambiguities in specification and design

MBT Development Flow



Toolset Architecture



Model Specification Module

Capability:

- **Tabular entry of system requirements.**
- **Definition of the system boundary by itemizing all input stimuli and responses**
- **Specifying traceability via requirement tags.**
- **Enumeration of input stimulus sequences**
- **Automatic analysis of the completed enumeration to verify coverage and to construct the usage model.**
- **Define usage variables and associate a unique set with each state in the model.**
- **Assigning probabilities to each transition in the usage model.**
- **XML schema for storing and managing the above data**

Test Generation and Analysis Module.

Capability:

- **Provides Markov analysis of the usage model for properties useful for model validation and test planning.**
- **Enables test case generation via random walk, relative probability, and graph coverage algorithm.**
- **Enables test case management necessary for pass/fail recording and format conversion.**
- **Provides analysis of test results to compute coverage and reliability metrics**

Test Translation Module

Capability:

- **Accepts operator input to build script fragments for each system stimulus and export the result to the script library.**
- **Reads stimulus mapping information from the script fragment library that maps the stimuli used in the model to codes readable by the Test Execution Module.**
- **Determines proper code sequences to perform the test cases created by the Test Case Generator.**
- **Generates test scripts for the Test Execution Module from the fusion of script fragments**

Test Execution Module

Capability:

- **Executes target specific test scripts using hardware and software elements designed to interface with the system under test.**
- **Provides the operator an interface to observe the test steps being performed as well as enabling the operator to pause or restart testing.**
- **Logs any results generated from the testing in formats for human interpretation and for input to the Test Case Analysis and Generation Module**

Auto-Tester

Capability:

- **Perform end-to-end testing of System Software.**
- **Record scripts from a PC keyboard and play them back to the keyboard port of a PC.**
- **Translate the serial communication between the Display Unit (DU) and the AFCS Computer Unit (ACU).**
- **In order to support the Enhanced Display System (EDS), the connection to the Auto Tester would be inserted between FBCB2 and the ACU, not between the EDU and FBCB2**

Automated Test Capability

Capability:

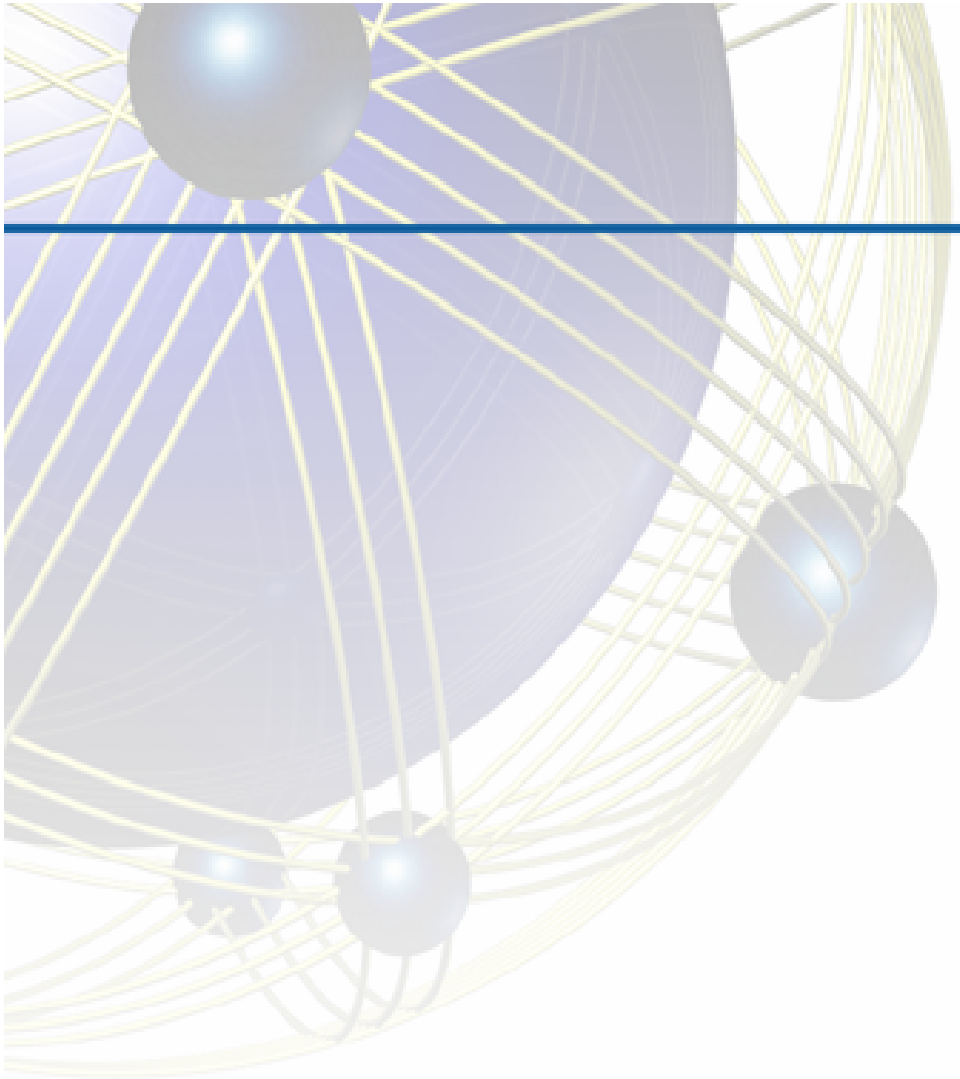
- **Supports Developmental, Integration, and Formal Qualification Testing (FQT) of a Fire Control Software System.**
- **Provides and demonstrates a means to capture test cases and procedures in a reusable form.**
- **Supports management of test artifacts, including storage, retrieval, editing, merging, and searching.**
- **Perform end-to-end testing of a Fire Control system software.**
- **Monitors and records the system's responses to stimulus, and, as necessary, emulates the appropriate response via a system interface to complete a given test case.**

Applying MST to Achieve Software Safety

- Traditional approaches include static analysis
- MST provides a robust, dynamic approach
  - ✓ Models cover all usage states, including rare ones.
  - ✓ Statistical testing ensures that potentially hazardous unknown or unforeseen events are covered in the system test suite. Static analysis alone cannot predict the consequences of highly complex behaviors.
- MST is a supplement to, not a replacement for, methods such as Fault Tree Analysis and Hazard Analysis.

Summary

- Automated Software Testing Increases Test Quality and Coverage Resulting in Improved Software Reliability.
- Project starts FY06
- Results will be provided in a final report and demonstration.
- Advance the state of the practice for system level MST.
  - Create large models of complex system software behaviors that closely represent expected operational behavior of a specific system.
  - Automatically generate test cases from the model.
  - Define and store test scripts associated with every stimulus in the test population.
  - Generate executable test scripts.
- Integrated Suite of Tools.



Questions?