JPL Software Engineering Technology Capability Needs

Dr. Matthew Barry
Deputy Manager, IT Program Office
Manager, Software Engineering Technology, Space Mission Information Technology Office

Capability Needs
Scope

What's In	What's Out
Processes and Policies	Business systems
People and Roles	PI-specific science software
Practices and Principles	Outreach
Technology and Infrastructure	
Flight systems	
Ground systems	
System engineering integration	
Test and simulation systems	
Software dev/test tools	
Developed internally	
Acquired externally	
Software project management	

Capability Needs
Software Development Within JPL

[redacted]

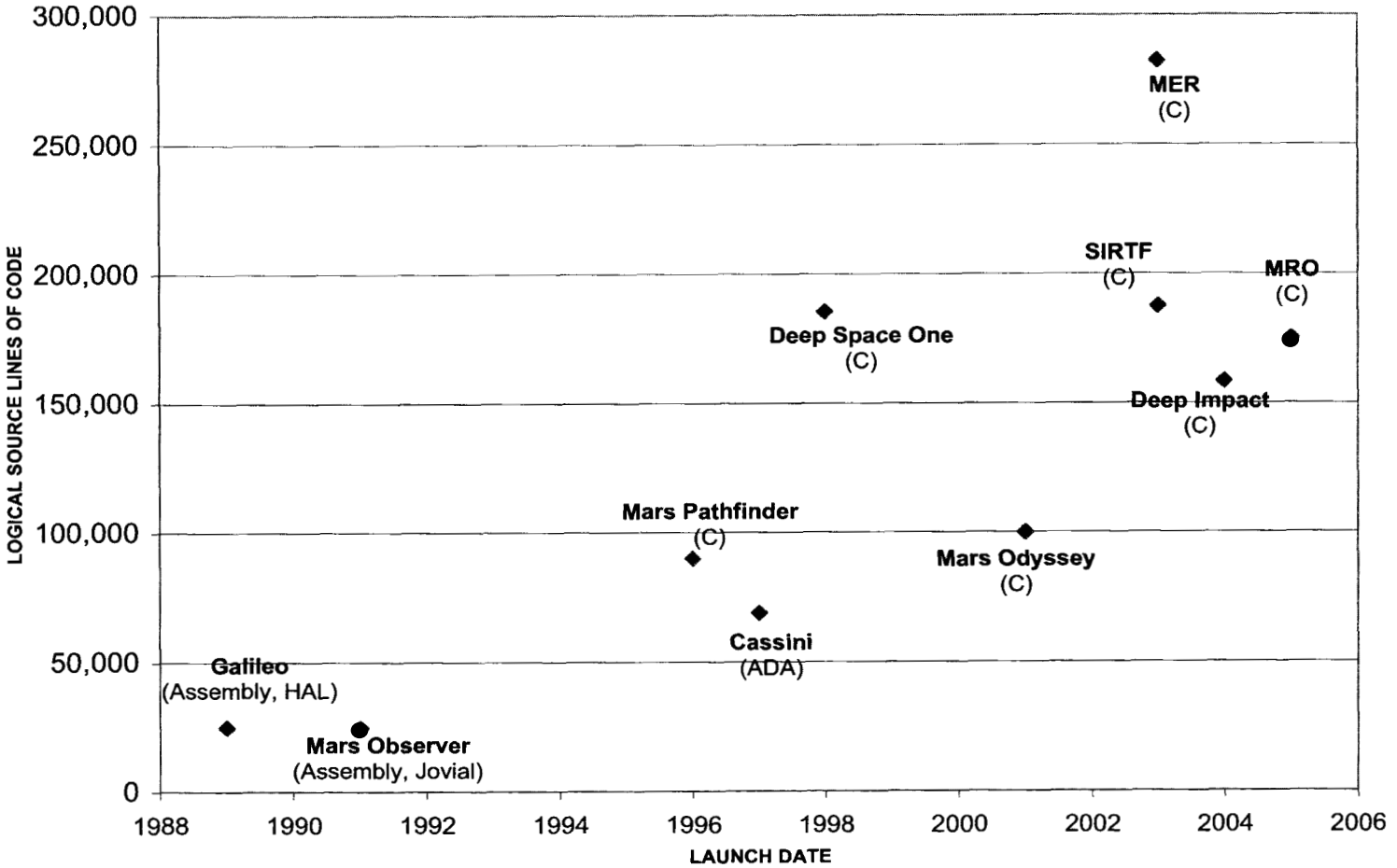
Initial CMMI Sections

Systems	Earth and Space Science	Telecommunications Science & Engineering	Avionics Systems & technology	Mechanical Systems & Engineering Research	Information Technologies & Software Systems	Observational Systems
Navigation **Mission Design** **Sequencing** **S/C Analysis** Science Planning Project Info Systems	Science Planning Science Processing Science Alg Dvlp	**GPS** **Embedded T-comm** Radar Processing	**C&DH** **Attitude Control** **Fault Protection** **Sim & Testbed** **Mobility** GSE		**DSN Mon & Control** **Command Processing** **Telemetry Processing** **Engineering Data Mgt** Enterprise Applications	**Instrument FSW** Science Processing Science Data Mgt Calibration/GSE

Created by SQI project

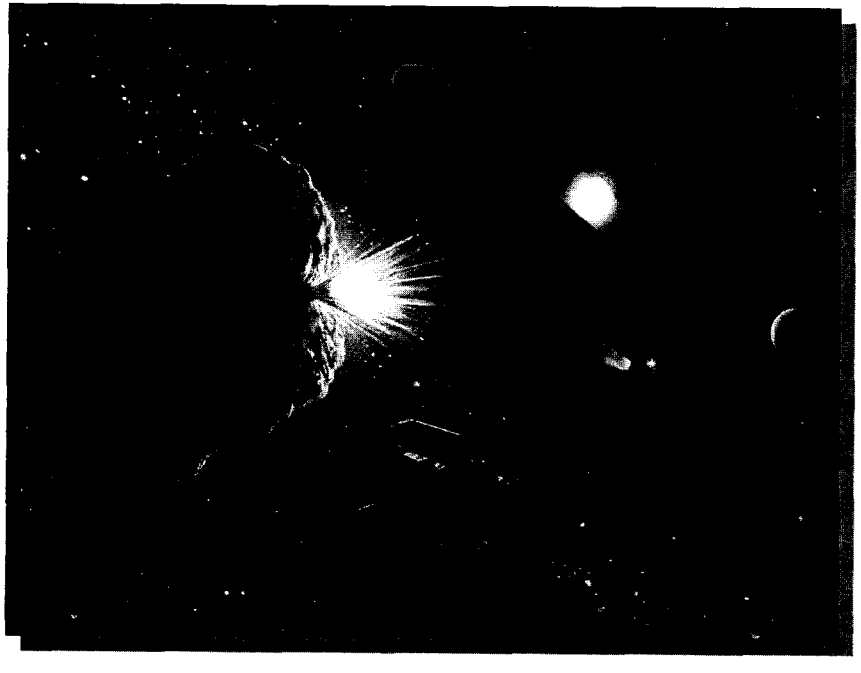
JPL Software Engineering Technology

Capability Needs
Flight Software Growth



Nichols TDAB 10/2003

Capability Needs
Deep Impact (DI)

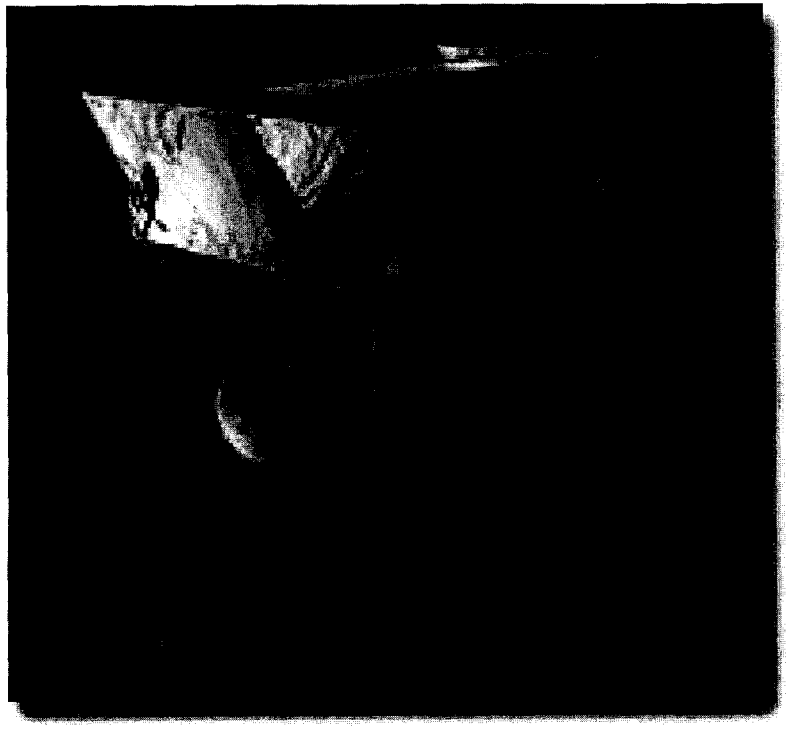


- Flyby and impactor spacecraft to excavate and study primordial material from a comet
- Unique autonomous navigation challenge to guide impactor
- Fault protection engine implemented with auto-generation of fault detection and response code
- Strong partnering with contractor on FSW

Capability Needs
Mars Science Laboratory (MSL)

- Unprecedented challenges for precision landing and surface operations on Mars
- Possible use of active hazard avoidance during descent
- Possible use of enhanced instrument placement capability
- Strong emphasis on simulation-based testing
- Autonomy possibly a driver for new system validation approaches
- Extended surface operations a driver for new operations concepts

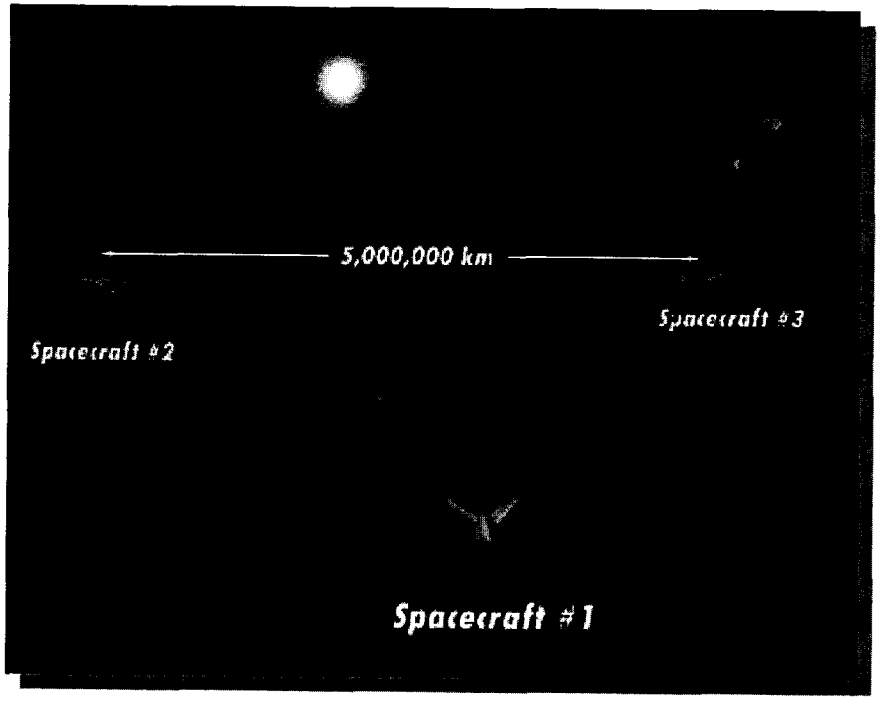
Capability Needs
Space Interferometry Mission (SIM)



- Deep-space interferometer to search for extra-solar planets
- Unprecedented precision requirements for controlling an instrument
- Unprecedented performance requirements for a real-time control loop
- Spacecraft fairly conventional

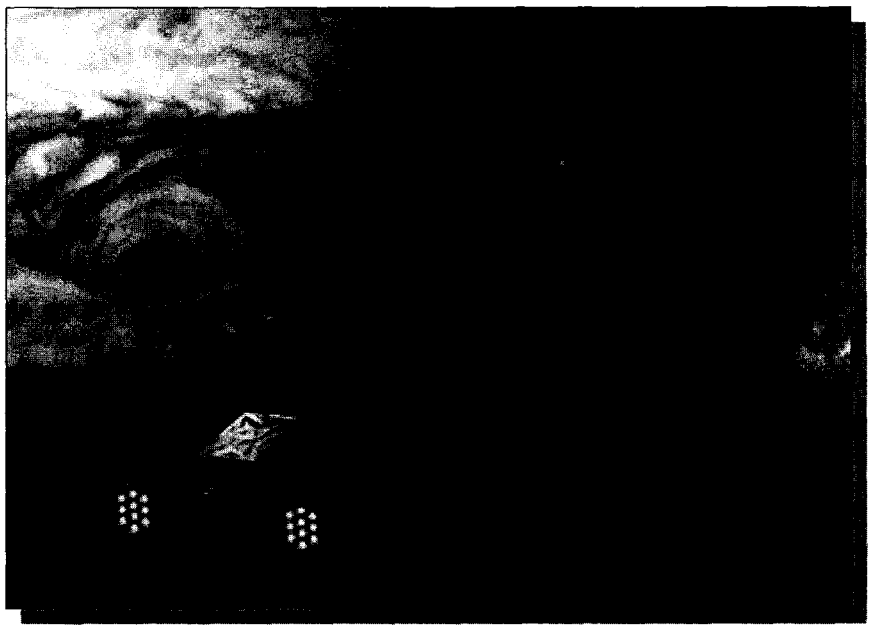
Capability Needs

Laser Interferometry Space Antenna (LISA)



- Joint GSFC-JPL mission
- Spacecraft in free fall around reference masses - perturbation sources carefully modeled - residual signal amounts to a gravity wave detection
- Initial formation creation requires sophisticated control approach
- Autonomous operations and fault protection needed to avoid data outages

Capability Needs
Jupiter Icy Moons Orbiter (JIMO)



- New propulsion and power concepts enable moon-hopping (Europa, Ganymede, Callisto)
- "Missions within a mission" highlights the need for a design approach to reusable software

Capability Needs
Expectations Assessment

Mission Paradigms and Support Trends

Level of Increasing Software Role

Mission / Operations Paradigms	Observatories situated farther from Earth	Constellations of small, low-cost spacecraft	Detailed orbital remote sensing	*In Situ* exploration via long-lived mobile elements	Longer and more concurrent missions
More onboard autonomy, less low-level commanding					
Frequent software uploads or adaptations					
More data acquisition, on-board processing and dissemination					
Increasingly complex flight-ground interactions					

IND JPL Customer Set analysis (prior to POTUS announcement 1/14):

Flight Software Problem Root Causes

- Requirements late
- Requirements inadequately specified for software implementation
- Late and inadequate testbed resources
- Inadequate time for testing
- Lack of software engineering perspective in project and flight systems engineering
- Flight and ground software not integrated
- Software complexity increasing while schedules becoming shorter
- Lack of a reusable code base for flight software and simulations
- Distribution of software development across multiple organizations
- Lack of contractor oversight

Capability Needs
Gap Analysis

- **Prior issues and observations (*where we are*):**
  - Respond to results of software problem root cause study
  - Achieve consistent across-the-lab software measurement definition, collection, lessons, and utilization
  - Formulate and apply adequate test strategies
  - Plan for software obsolescence
  - Plan software projects to avoid budget and schedule pressure
- **New issues based on future expectations (*where we need to be*):**
  - Software and systems handling substantially more complexity
  - Increased need for software training for those without requisite skills
  - Demonstrable product dependability for risk arbitration decisions
  - Prevention or earlier detection of defects
  - Mission element interdependence demands more analytic capabilities for software architecture design
  - Planned re-use and product lines; incremental functionality

JPL Software Engineering Technology

Capability Needs
Challenge Summary

Challenge	Proposed Solution	Benefit
Demonstrating software process improvement	•Create, apply, and mandate uniform set of measurements within each software domain •Create and deploy corresponding collection and analysis tools •Collect and act upon lessons	•Enables objective process optimization tailored to each community •Anticipates CMMI-3,4,5
Coordinating software engineering R&D efforts, leveraging sponsorship opportunities	•Establish and manage software R&D portfolio and pipeline •Support establishment of HQ software engineering practice	•Better coordination of few software R&D opportunities •Increased infusion
Increasingly complex software becoming more vital to mission success, and more central to mission risk management	•Double size of LaRS to accelerate technology deliveries and infusion •Sponsor industry forum developing mission-critical dependability benchmarks	•Quality designed-in early in the lifecycle, reducing pass-through defects •Demonstrable risk posture for project decision-making
Software engineering skills are becoming increasingly critical; need to know what we know, while skill levels vary widely	•Establish under SQI a training and certification program for software engineering skills •Require mission-critical projects to use certified software team members	•Higher quality and confidence in project team members •Higher consistency in demonstrable skills •Higher software quality

Capability Needs

Current Initiatives Addressing Gaps

- **Software Quality Improvement (SQI) Project**
  - Preparing community for process-centric development
  - Mapping and implementation of key practice areas
  - Training in and infusion of best practices
- **Mission Data System (MDS) Project**
  - Bridging the gap between systems and software engineering
  - Implementing component-based framework
  - Intended for planned re-use
- **Applied research efforts**
  - **Laboratory for Reliable Software (LaRS)**
    - R&D toward achieving reliability in mission-critical software
  - **Software Techniques and Methodologies (R&TD)**
    - New platforms, languages, and engineering techniques
  - **Software Assurance Research**
    - Techniques for requirements tracing, V&V of autonomy, contingency planning for autonomy, software security

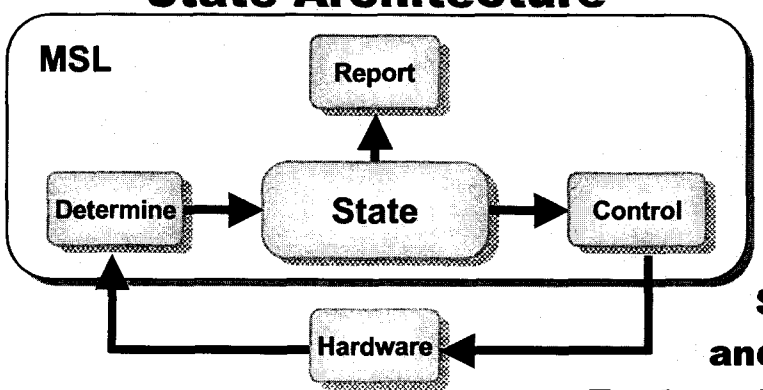
Capability Needs
Software Quality Improvement

- ***Establish JPL mission software development, cost, schedule, risk, quality and acquisition practices as equal to the best in industry***
- **Commitment**
  - JPL is committed to the goal of becoming a CMMI Level 3 organization for mission-critical software
- **Strategy**
  - Consistent with the advice from experienced organizations and process improvement leaders, focus on a core group of committed line managers
    - Navigation (312); Flight Software (345); Command and Telemetry (369); Instrument Flight Software (382)
- **Goals**
  - Step 1: Four software-intensive sections achieve Level 2± by the end of FY05
  - Step 2: Four software-intensive sections achieve Level 3± by the end of FY07
  - Step 3: The remainder of the sections producing mission-critical software achieve Level 3 by the end of FY09

Capability Needs
Mission Data System

Integrate best software engineering and systems engineering practices

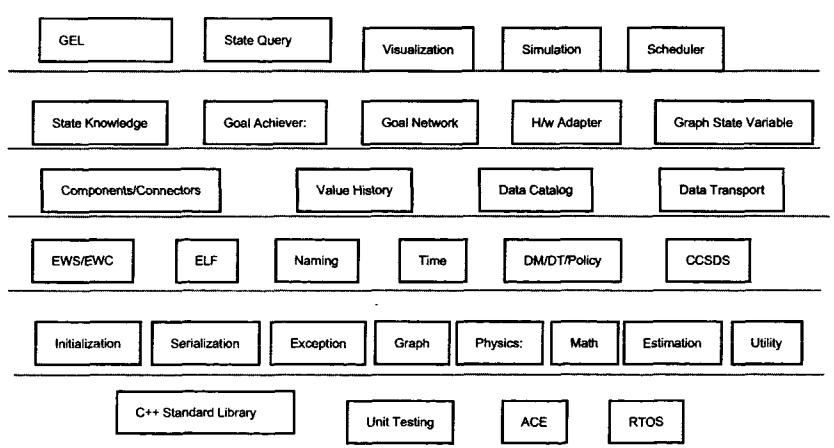
State Architecture



State Analysis requirements map directly to software

Systems and Software Engineering use the same vocabulary

No translation, consistent representation



Reusable Frameworks

Capability Needs
Laboratory for Reliable Software

- **Responds to "Reliable Software" needs and opportunities**
  - Mission critical software becoming more complex, testing is inadequate
  - Project Managers need software reliability insight in order to manage software risk
  - Significant investment in reliable software in several industries (aerospace, communications, automotive) has led to rapid maturity of tools and techniques
  - Provides lab focus for software reliability R&D, acquisition, and insertion
  - Focuses on technology, complementing SQI's focus on process
- **Identifies specific areas of concern**
  - Software requirements
  - Software design
  - Software metrics
  - Source code analysis
- **Identifies key application target areas**
  - Autonomous systems
  - Fault protection systems
  - Fault-tolerant software systems

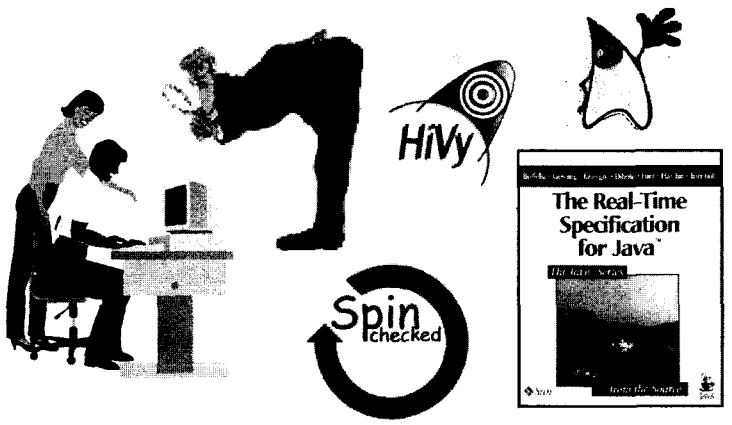
Capability Needs
Software Assurance Research

- **The IV&V Facility manages the Office of Safety and Mission Assurance Software Assurance Research Program**
  - The OSMA sponsors research initiatives from NASA Centers, West Virginia University, other universities, and industry
  - JPL, through the Assurance Technology Program Office, has for many years been a recipient of funding from this source for software assurance related research.
- **FY'04 tasks:**
  - Requirements Decomposition Analysis - aim: better able to assure the correctness of requirements flow down (where requirements at level N trace to requirements at level N+1)
  - V&V of autonomy - application of model checking techniques to autonomy software
  - Contingency measures for autonomous software
  - Metrics-bases analysis of the relationship between requirements churn and defects
  - Software Security - checklists for, tools for, contingency for autonomy

Capability Needs

R&TD Software Techniques & Methodologies

Increase mission software reliability and reuse



Objective

- Research, develop, apply and evaluate leading software engineering tools, techniques and methodologies
  - Evaluate real-time Java against FSW demands
  - Implement model checking for the validation of mission-critical flight software
  - Develop a robust infrastructure for applying the techniques of evolvable computation to JPL engineering problems
  - Develop techniqies to enable static analysis of C++

Tasks

- A New Flight Software Platform Based on Real-time Java and Real-time Linux
  - PI: Dr. Daniel Dvorak
- Rapid Adoption of Model-based Validation for Mission-Critical Flight Software
  - PI: Paula Pingree
- Evolvable Computation Technologies for Space Systems
  - PI: Dr. Rich Terrile
- Verifiable C++
  - PI: Kirk Reinholtz

Benefits

- Relative to C++, Java saves money on development and maintenance and is safer due to automatic memory management
- The power of formal methods upgrades traditional testing for increasingly complex systems with critical reliance on software
- Help systems engineers explore significantly increased volumes of design space and to quickly optimize designs
- Introduce rigor into C++ coding practices not enforced by the language itself

Capability Needs
Desired SET Program Content

- Mission-critical software assuredness measures
- Software product line analysis
- Model-based development, analysis and execution
- Statistical process and product analysis
- Formal and functional specification and implementation
- Composable frameworks, verification and certification
- Simulator and testbed trade analytics
- Integrated system and software engineering

Capability Needs

End