

Systems and Software Design Principles for Large-Scale Mission-Critical Embedded Products from Aerospace and Financial Problem Domains

October 2008

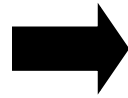
Rick Selby

**Head of Software Products
Northrop Grumman Space Technology
310-813-5570, Rick.Selby@NGC.com**

**Adjunct Professor of Computer Science
University of Southern California**

Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

- 
- **Systems and software engineering strategies, principles, benefits, and tradeoffs**
 - **Example large-scale mission-critical embedded software system**
 - **Investigations of synthesis, analysis, and modeling principles**
 - Synthesis: Lifecycle models
 - Synthesis: System architectures
 - Analysis: Reuse analysis
 - Analysis: Structure analysis
 - Modeling: Defect detection techniques
 - Modeling: Measurement and prediction
 - **Conclusions and future work**

Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

BENEFITS and TRADEOFFS

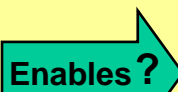
SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases



- Organization of and parallelization within large-scale projects
- Rapid feedback and innovation
- Visibility into stabilization and handoffs

- System architectures: Layered system architectures containing embedded meta-language programs and interpreters



- User-customizability
- Multi-platform portability
- Automated testing

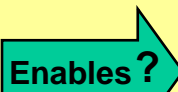
ANALYSIS

- Reuse analysis: Reconfigurable component-driven development



- Sustainable multi-project reuse
- Lower component defect rates
- Lower component development effort

- Structure analysis: Inter-component connectivity analysis



- Lower component defect rates
- Lower component defect correction effort
- Lower component development effort

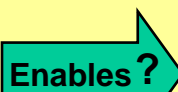
MODELING

- Defect detection techniques: Disciplined team-based peer reviews



- Early lifecycle defect detection
- Low out-of-phase defect rates
- High return-on-investment for prevention

- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models



- Early identification of high defect or high effort components
- Statistical process control
- Pro-active process guidance

Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

- **Systems and software engineering strategies, principles, benefits, and tradeoffs**

- 
- **Example large-scale mission-critical embedded software system**

- **Investigations of synthesis, analysis, and modeling principles**

- Synthesis: Lifecycle models
- Synthesis: System architectures
- Analysis: Reuse analysis
- Analysis: Structure analysis
- Modeling: Defect detection techniques
- Modeling: Measurement and prediction

- **Conclusions and future work**

Organizational Charter Focuses on Embedded Software Products

- Embedded software for
 - Advanced robotic spacecraft platforms
 - High-bandwidth satellite payloads
 - High-power laser systems
- Emphasis on both system management and payload software
- Reusable, reconfigurable software architectures and components
- Languages: O-O to C to asm
- CMMI Level 5 for Software in February 2004; ISO/AS9100; Six Sigma
- High-reliability, long-life, real-time embedded software systems

Software Development Lab



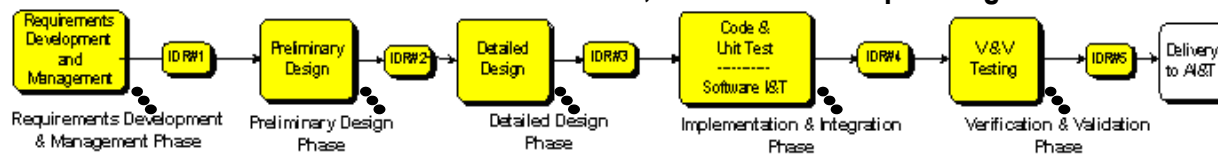
Software Analysis



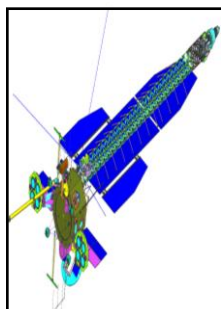
Software Peer Review



Software Process Flow for Each Build, with 3-15 Builds per Program



Prometheus / JIMO



GeoLITE



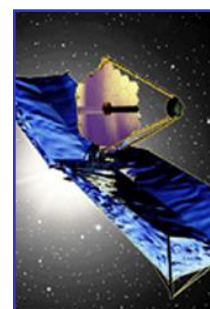
NPOESS



AEHF



JWST



MTHEL



EOS Aqua/Aura



Airborne Laser



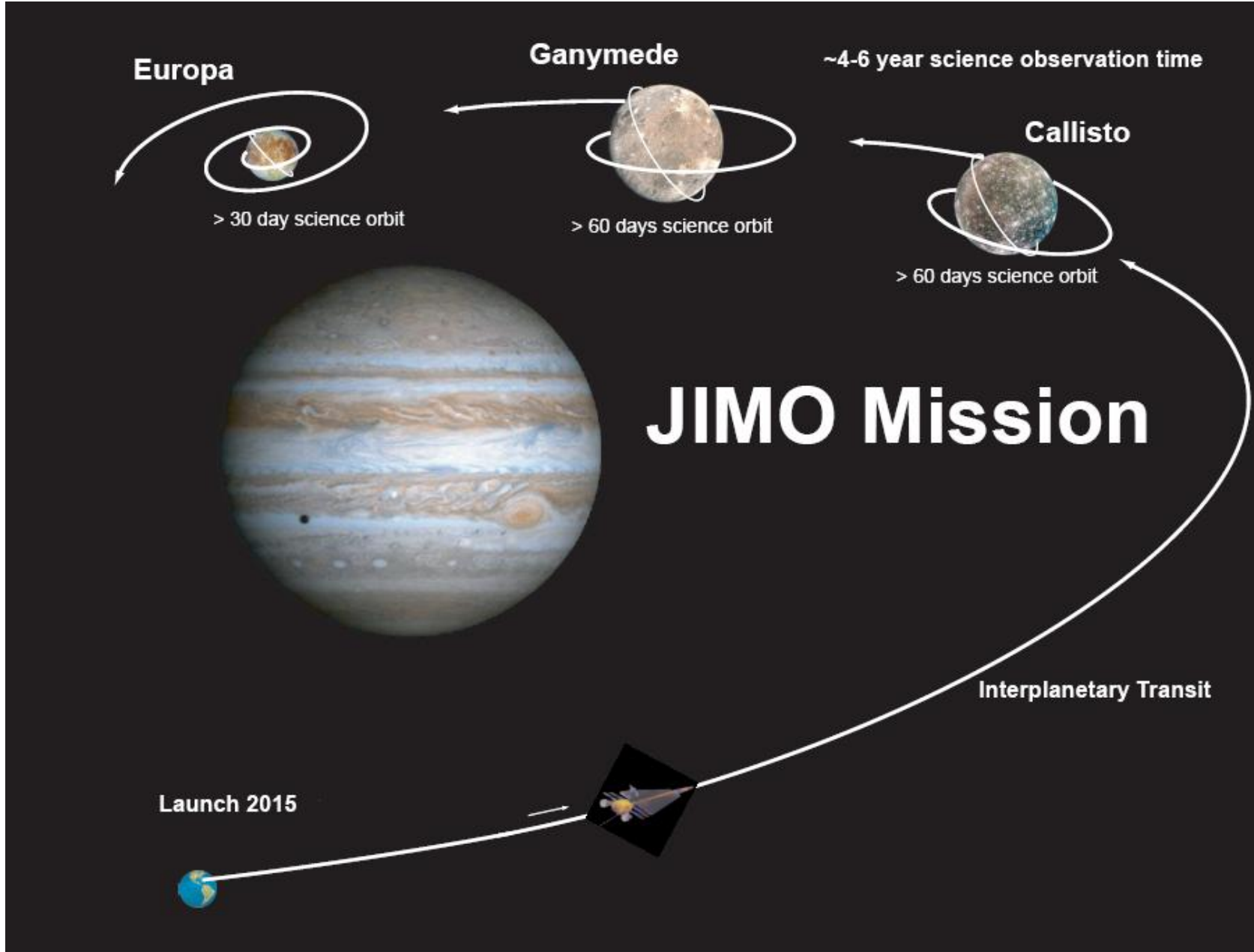
Chandra



Restricted

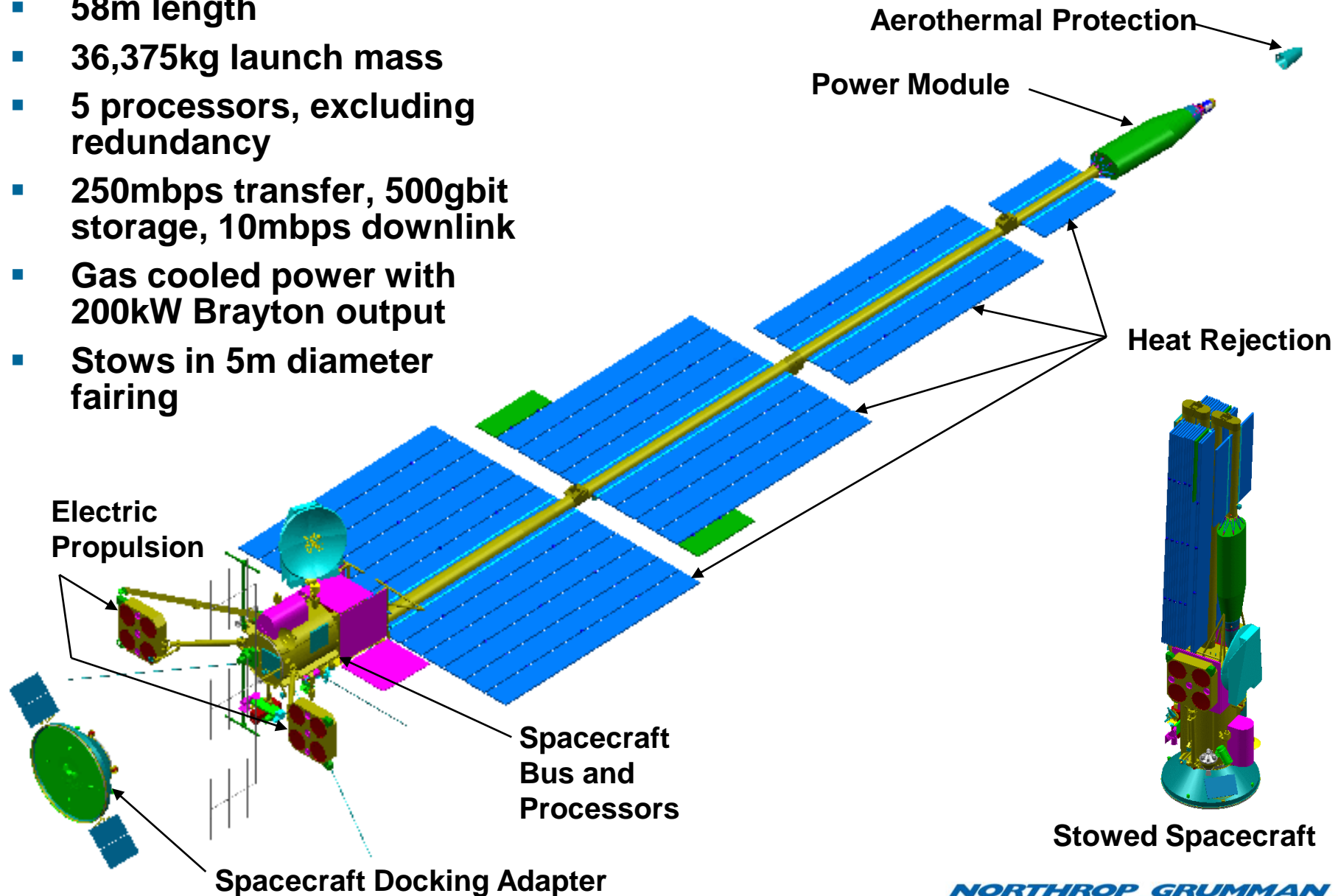


Prometheus Spacecraft Supports Jupiter JIMO Mission over 9 to 14 Year Duration



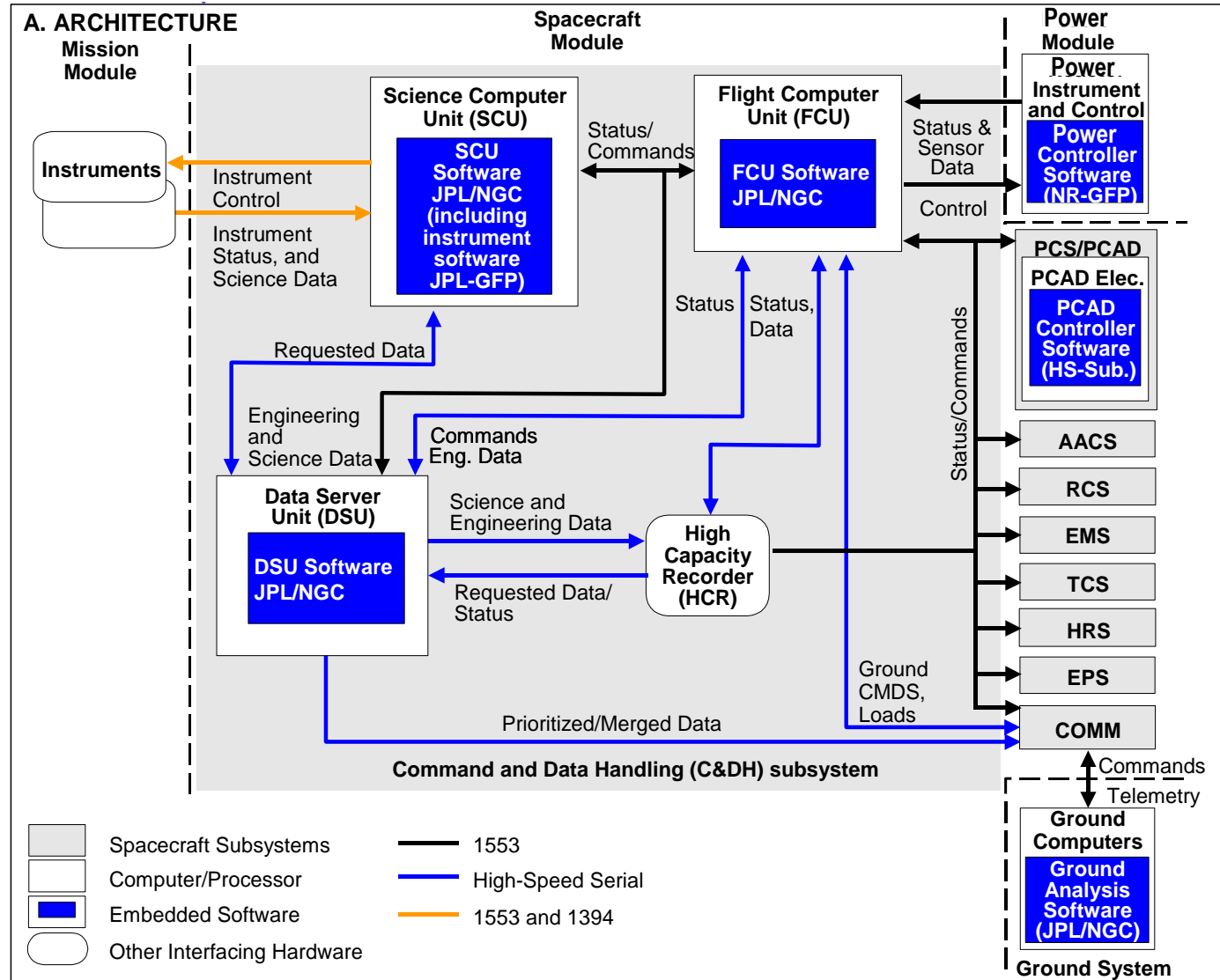
Prometheus Spacecraft for JIMO and Related Missions Enables Data-Intensive Science

- Spacecraft configuration PB1
 - 58m length
 - 36,375kg launch mass
 - 5 processors, excluding redundancy
 - 250mbps transfer, 500gbit storage, 10mbps downlink
 - Gas cooled power with 200kW Brayton output
 - Stows in 5m diameter fairing



Architecture Defines 5 Processors: Flight, Science, Data, Power Generation, and Power Distribution

- Embedded software** implements functions for commands & telemetry, subsystem algorithms, instrument support, data management, and fault protection
- Size of on-board software growing to accelerate data processing and increase science yield
- Software “adds value” to mission by enabling post-delivery changes to expand capabilities and overcome hardware failures



Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

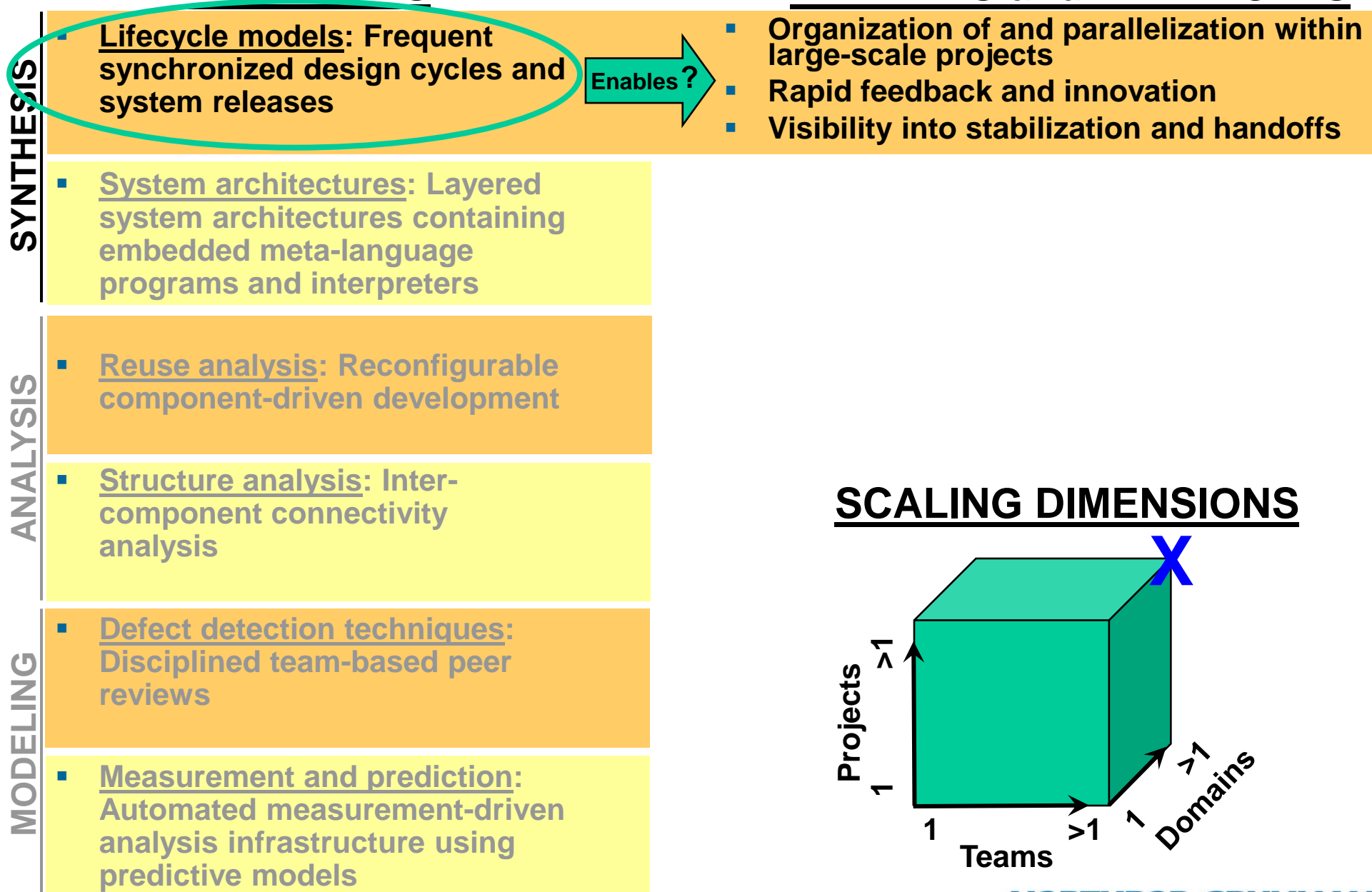
Overview

- **Systems and software engineering strategies, principles, benefits, and tradeoffs**
- **Example large-scale mission-critical embedded software system**
- ➔ **Investigations of synthesis, analysis, and modeling principles**
 - Synthesis: Lifecycle models
 - Synthesis: System architectures
 - Analysis: Reuse analysis
 - Analysis: Structure analysis
 - Modeling: Defect detection techniques
 - Modeling: Measurement and prediction
- **Conclusions and future work**

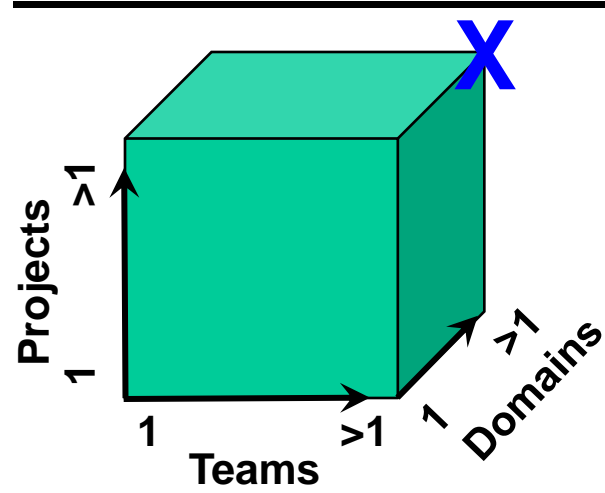
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

BENEFITS and TRADEOFFS



SCALING DIMENSIONS



Incremental Software Builds Deliver Early Capabilities and Accelerate Integration and Test

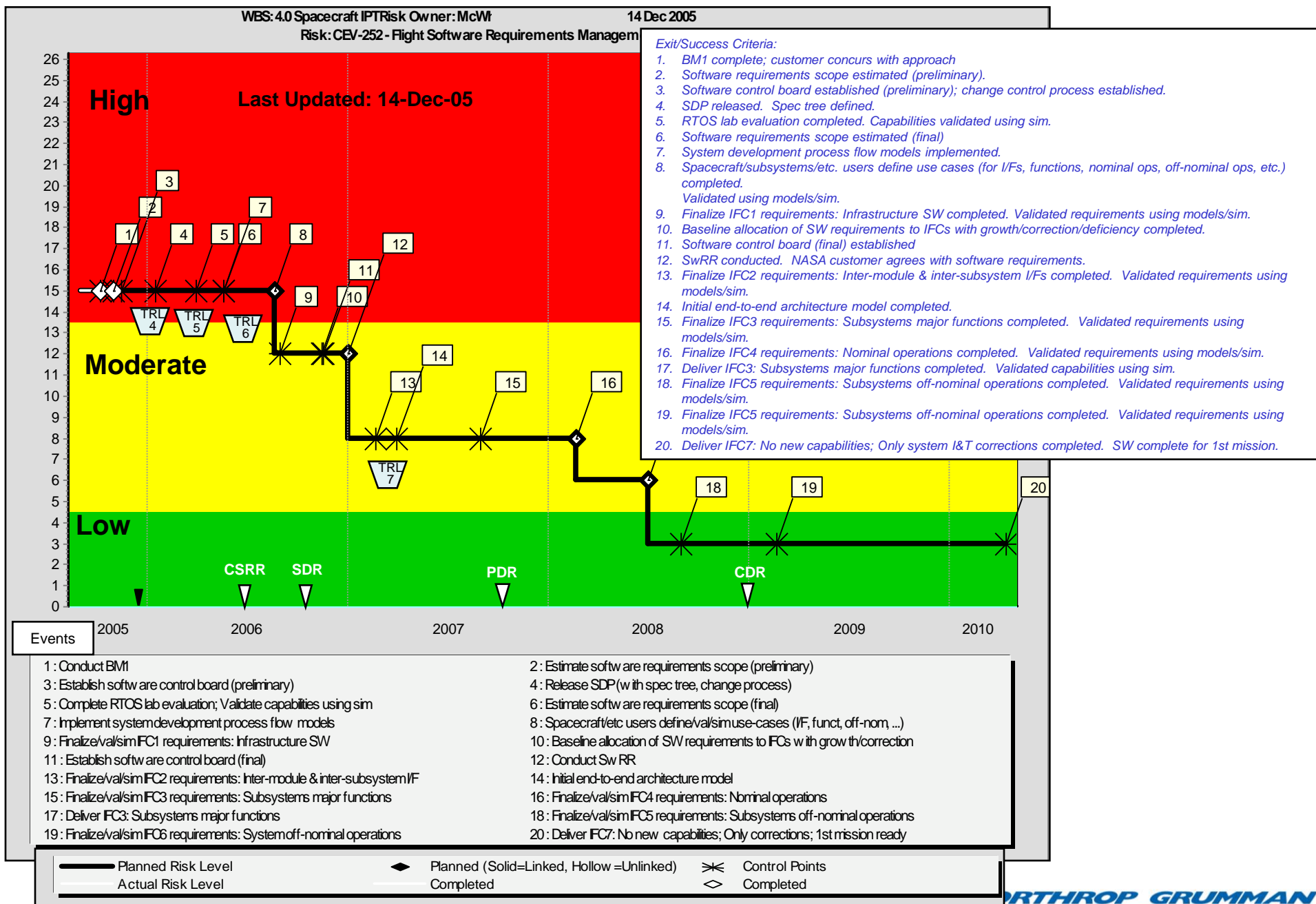
CY 2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	
A	B				C		D			
ATP△△	PMSR			SM PDR△		SM CDR△		BUS I&T△	SM AI&T△	Delivered to, Usage
11/04	1/05			6/08		8/10		8/12	8/13	
Flight Computer Unit (FCU) Builds										JPL/NGC, Prelim. Hardware/Software Integration
										JPL/NGC, Final Hardware /Software Integration
										JPL, Mission Module Integration
										NR, I Power Controller Integration
										NGC, AACS Validation on SMTB
										NGC, TCS/EPSS Validation on SSTB
										NGC, Fault Protection S/W Validation on SSTB
Science Computer Unit (SCU) Builds <i>Note: Science Computer builds for common software only (no instrument software included)</i>										JPL, Prelim. Hardware/Software Integration
										JPL, Final Hardware/Software Integration
Data Server Unit (DSU) Builds										NGC, Prelim. Hardware/Software Integration
										NGC, Final Hardware/Software Integration
										NGC, HCR Integration on SMTB
Ground Analysis Software (GAS) Computer Builds										JPL, Prelim. Integration into Ground System
										JPL, Final Integration into Ground System

Legend:

	=			=	Design Agent		N is defined as follows:
	=			=	N Performer of Activity N		1 Requirements
	=			=	JPL		2 Preliminary Design
				=	NGC		3 Detailed Design
				=	Role/activity shared by JPL and NGC		4 Code and Unit Test/Software Integration
							5 Verification and Validation

04S01176-4-108f_154

Projects Define Risk Mitigation "Burn Down" Charts with Specific Tasks and Exit Criteria



Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases

- System architectures: Layered system architectures containing embedded meta-language programs and interpreters

Enables?

BENEFITS and TRADEOFFS

- User-customizability
- Multi-platform portability
- Automated testing

ANALYSIS

- Reuse analysis: Reconfigurable component-driven development

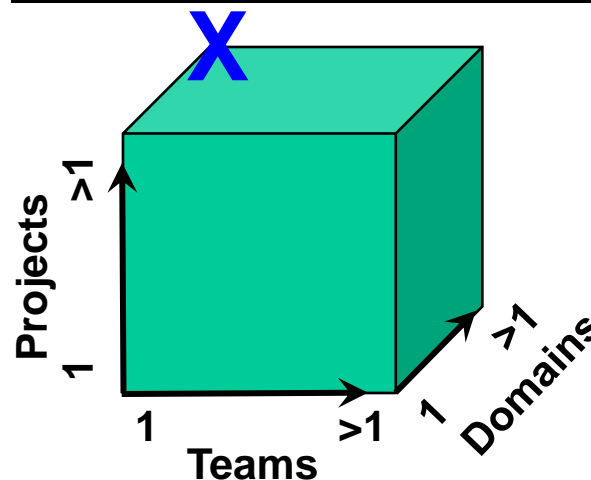
- Structure analysis: Inter-component connectivity analysis

MODELING

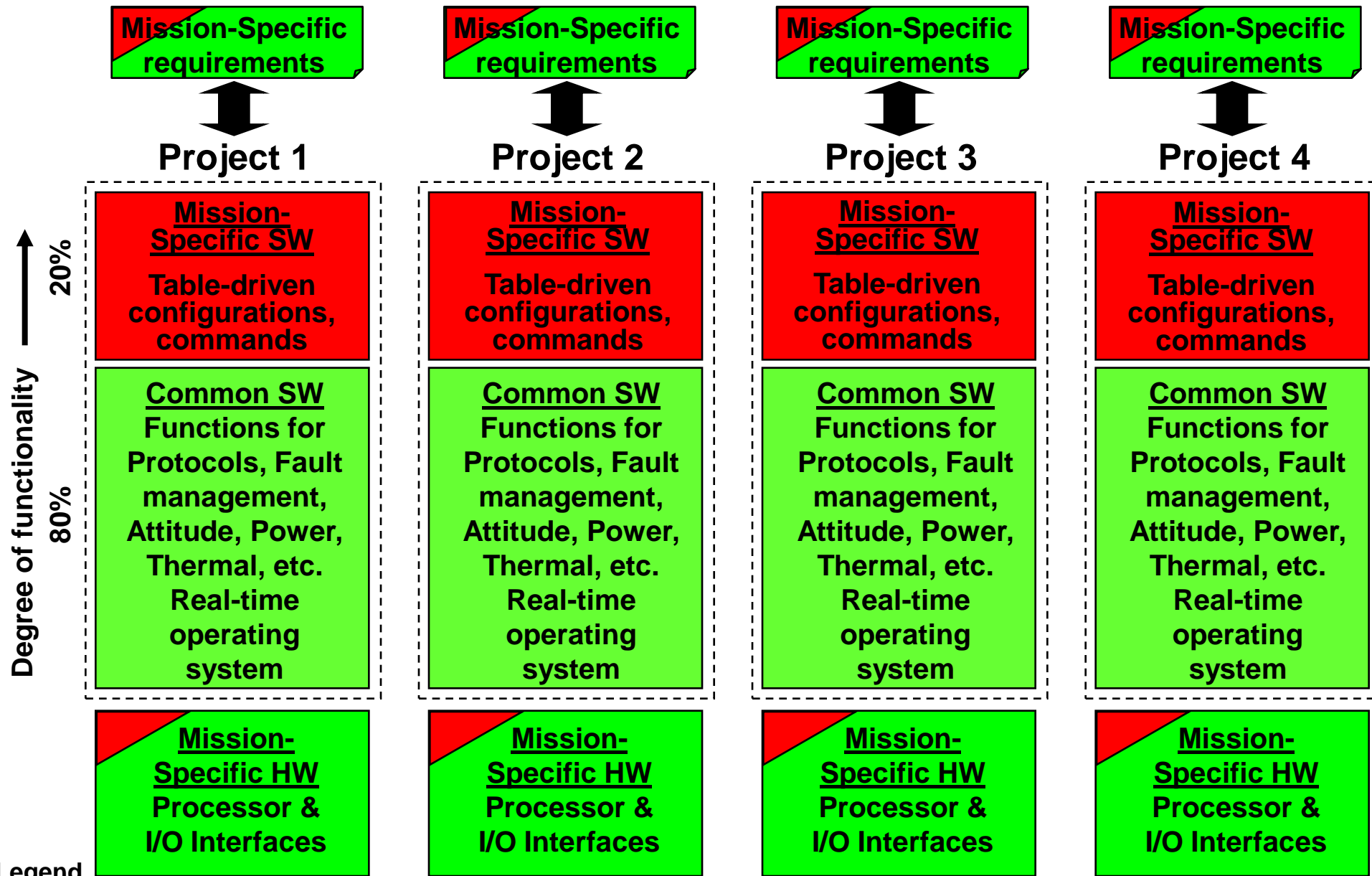
- Defect detection techniques: Disciplined team-based peer reviews

- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

SCALING DIMENSIONS



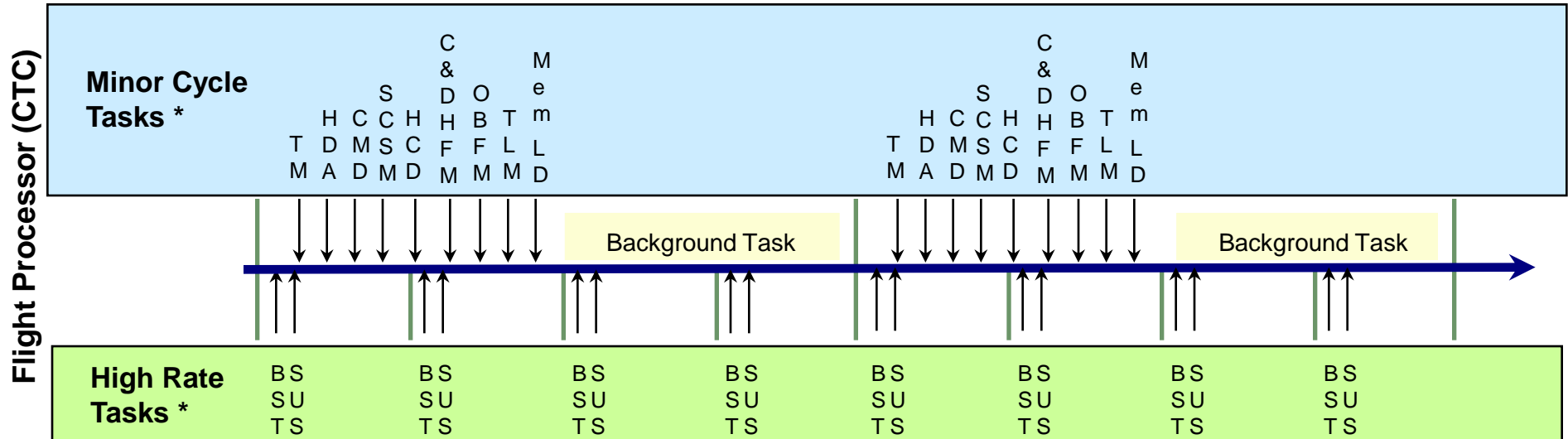
Common Requirements Enable Software Product Lines and Layered Architectures Across Projects



Legend

- Mission-specific
- Common across projects

Architecture Uses Simple Task Structure, Deterministic Processing, and Predictable Timeline



- Three-task structure: 32ms task (high rate), 128ms (minor cycle), and background task
- Minor cycle serves as the main workhorse task that executes commands, formats telemetry, and handles fault protection
- Minor cycle command processor reads active command sequences and executes individual deterministic commands
- >50% margins at system delivery for processor, memory, storage, and bus

* Not to scale

Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters

ANALYSIS

- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis

MODELING

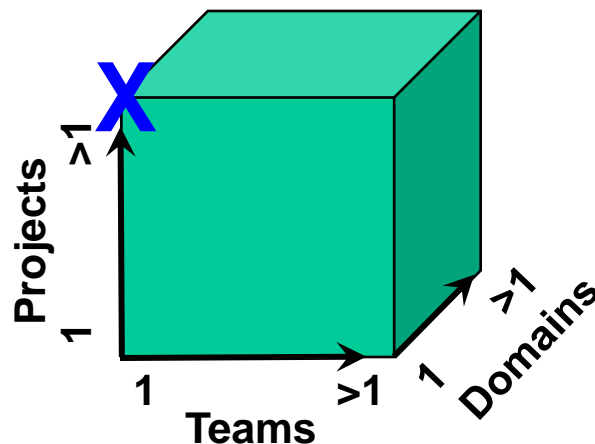
- Defect detection techniques: Disciplined team-based peer reviews
- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

Enables?

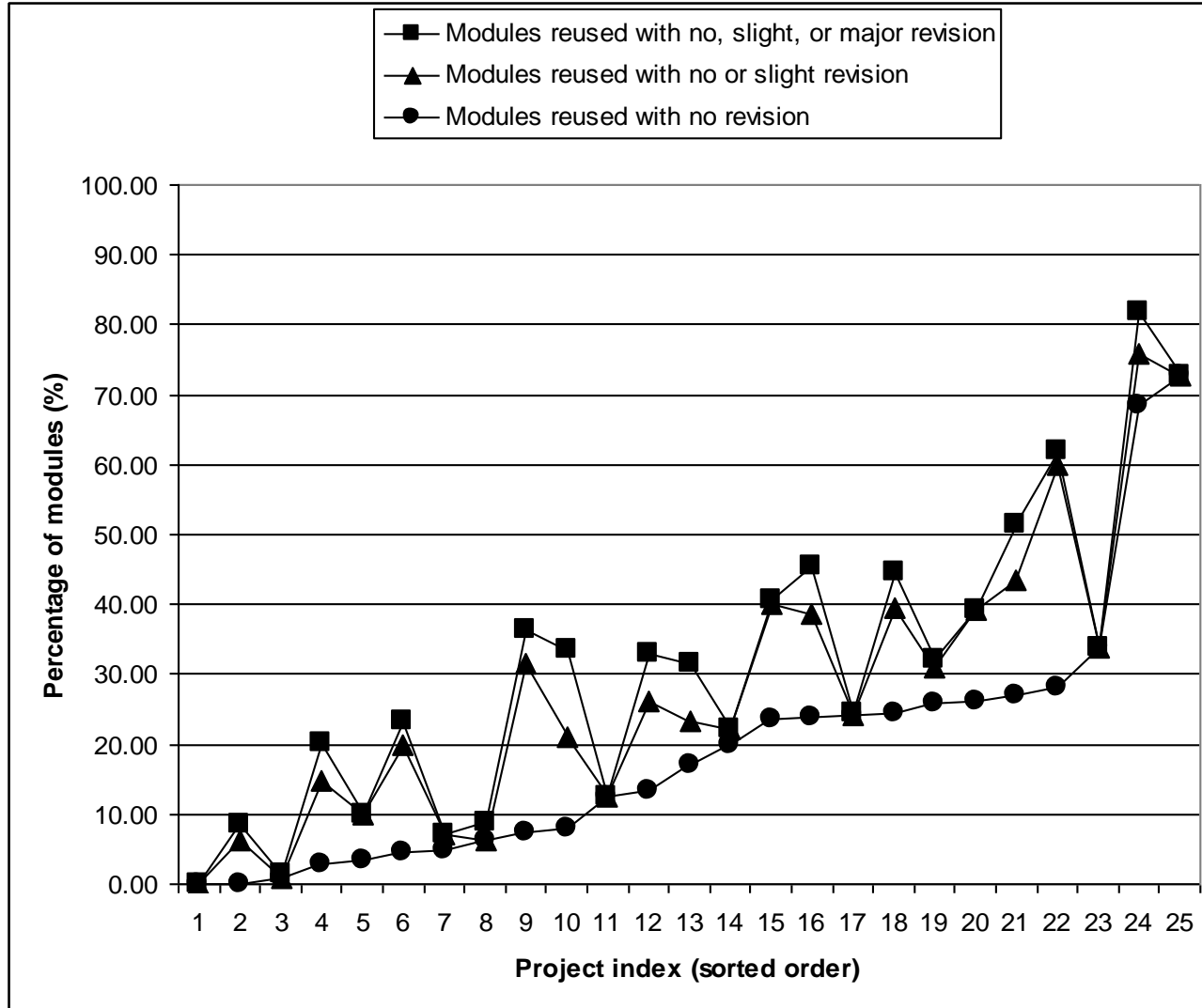
BENEFITS and TRADEOFFS

- Sustainable multi-project reuse
- Lower component defect rates
- Lower component development effort

SCALING DIMENSIONS

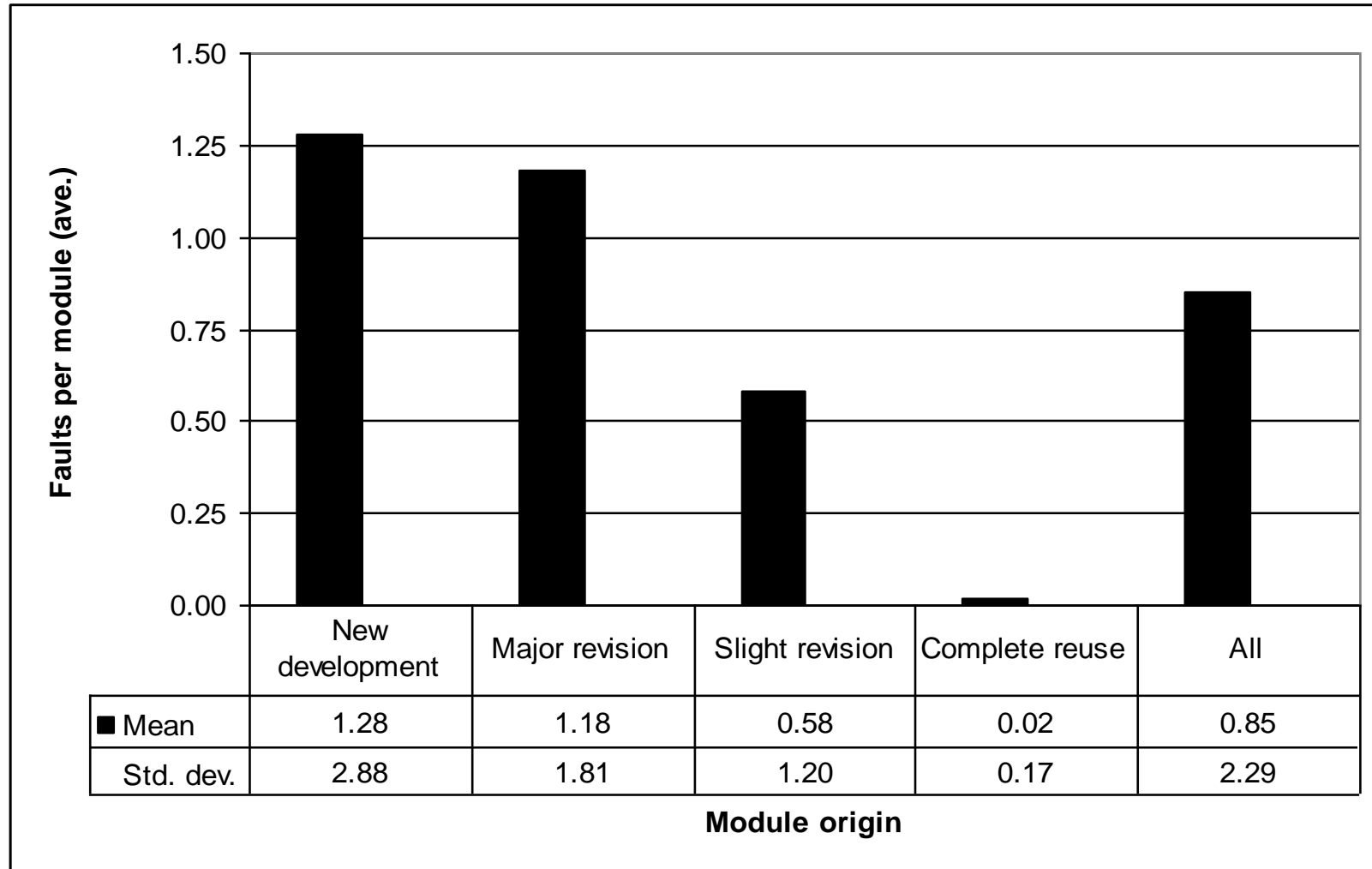


32% of Software Components are Either Reused or Modified from Previous Systems



- Data from 25 NASA systems
- Component origins: 68.0% new development, 4.6% major revision, 10.3% slight revision, and 17.1% complete reuse without revision

Analyses of Component-Based Software Reuse Shows Favorable Trends for Decreasing Faults



- Data from 25 NASA systems
- Overall difference is statistically significant ($\alpha < .0001$). Number of components (or modules) in each category is: 1629, 205, 300, 820, and 2954, respectively.

Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters

ANALYSIS

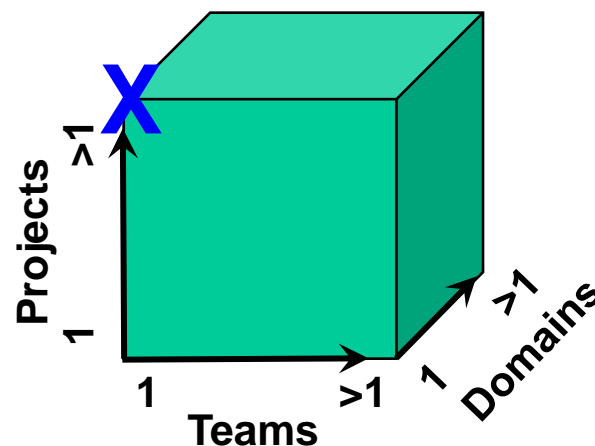
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis

Enables?

MODELING

- Defect detection techniques: Disciplined team-based peer reviews
- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

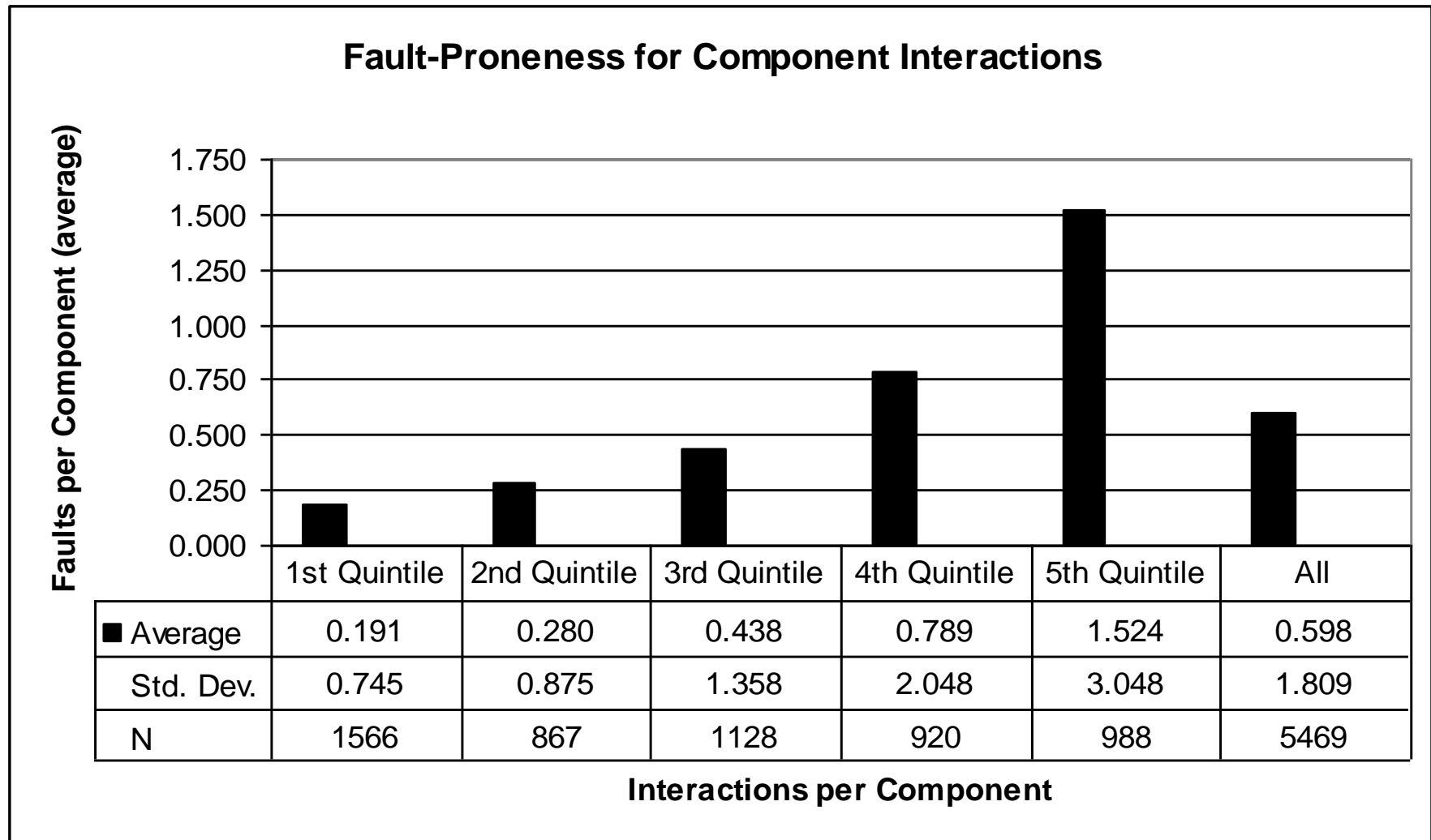
SCALING DIMENSIONS



BENEFITS and TRADEOFFS

- Lower component defect rates
- Lower component defect correction effort
- Lower component development effort

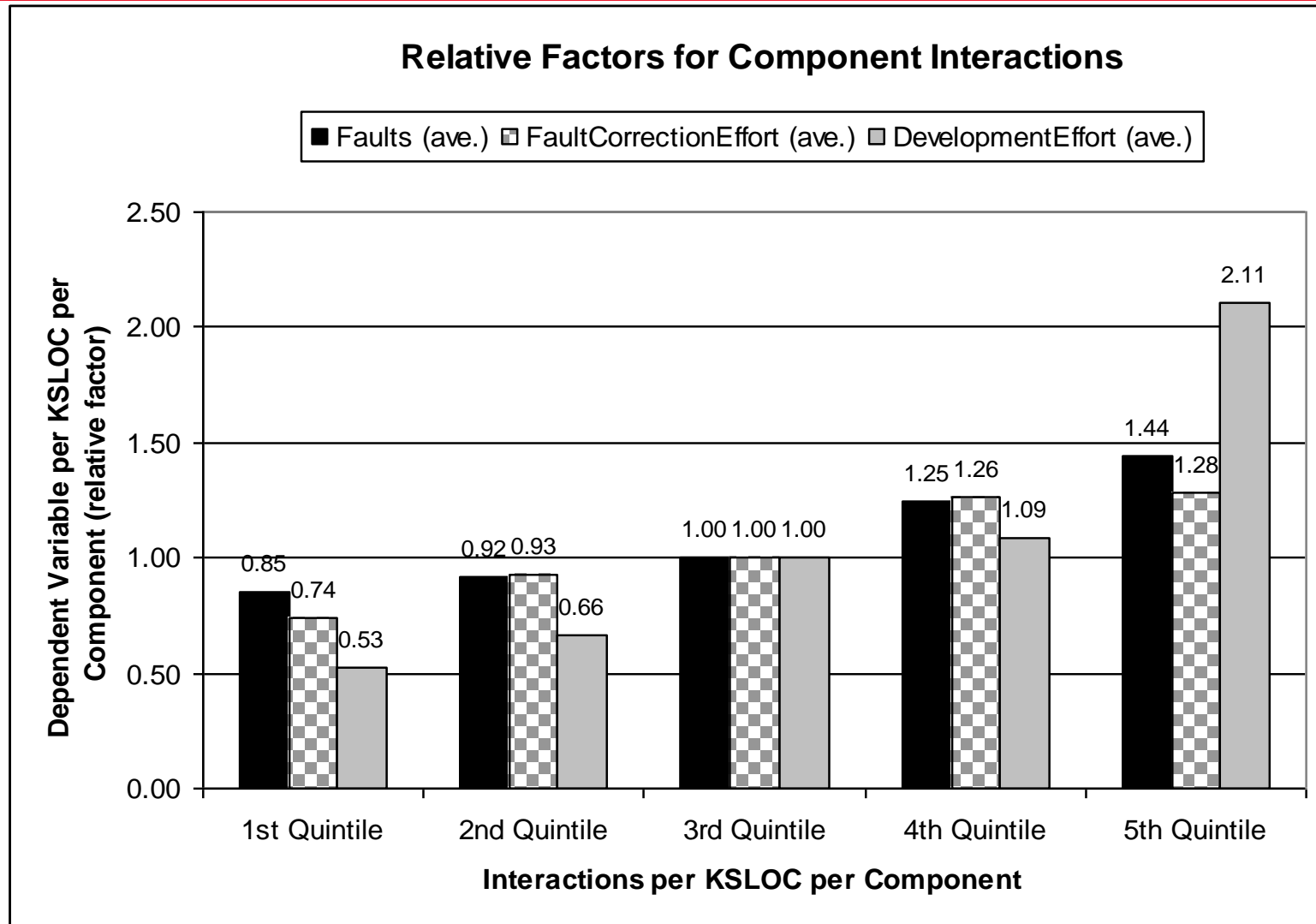
Analyses of Software Architectures Shows Fault Trends for Component Interactions



- Data from 23 NASA systems
- 5469 components analyzed and categorized by quintiles

Absolute

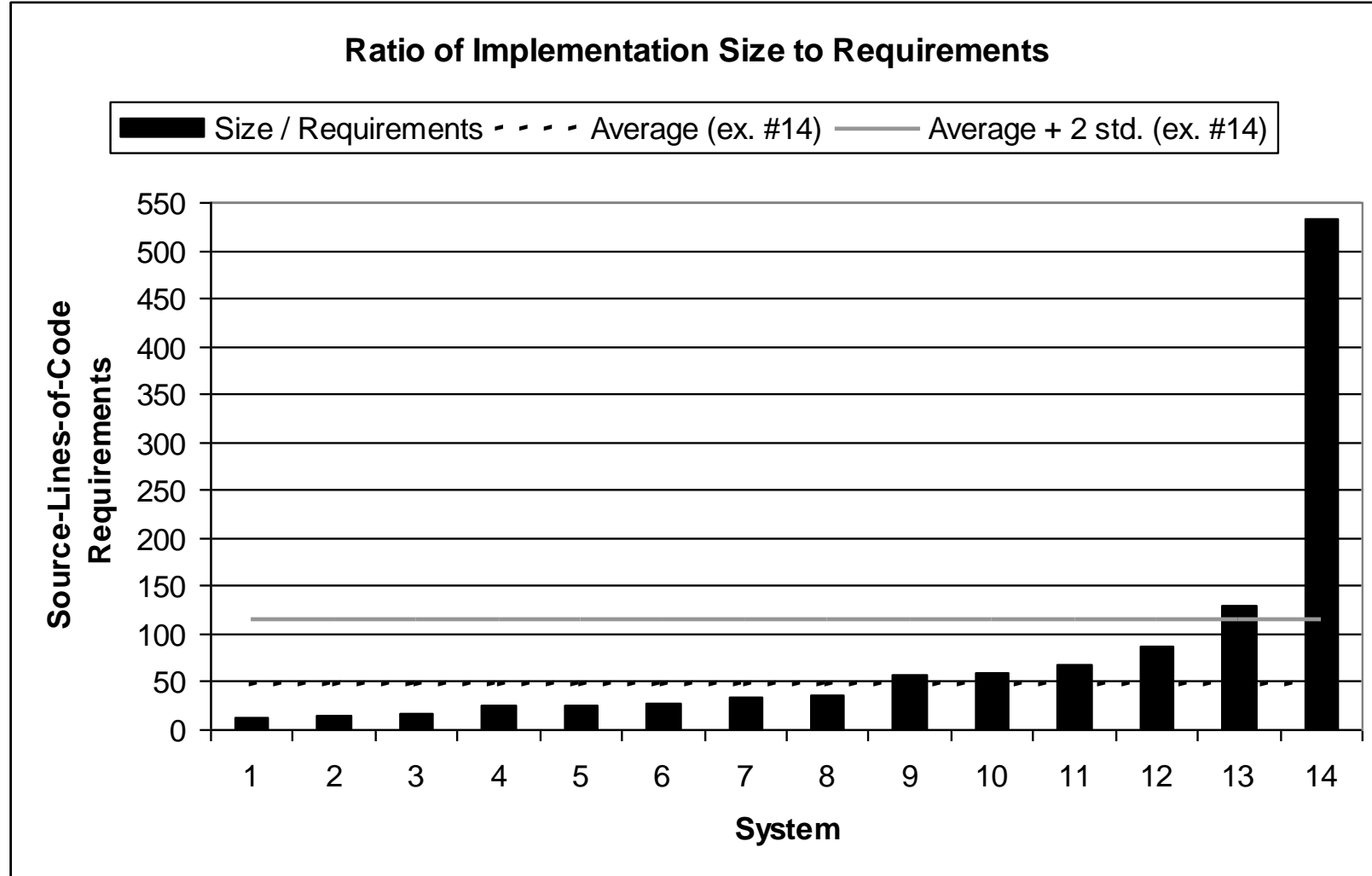
Analyses of Software Architectures Shows Fault Trends for Component Interaction Relative Factors



- Data from 23 NASA systems
- 5469 components analyzed and categorized by quintiles

Absolute norm-norm

Analyses of Software Requirements Shows Leading Indicators for Implementation Scope and Priorities



- Data from 14 NASA systems
- Ratio of implementation size to software requirements has 81:1 average and 35:1 median; Excluding system #14, the ratio has 46:1 average and 33:1 median
- Ratio of software requirements to system requirements has 6:1 average

Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases

- System architectures: Layered system architectures containing embedded meta-language programs and interpreters

ANALYSIS

- Reuse analysis: Reconfigurable component-driven development

- Structure analysis: Inter-component connectivity analysis

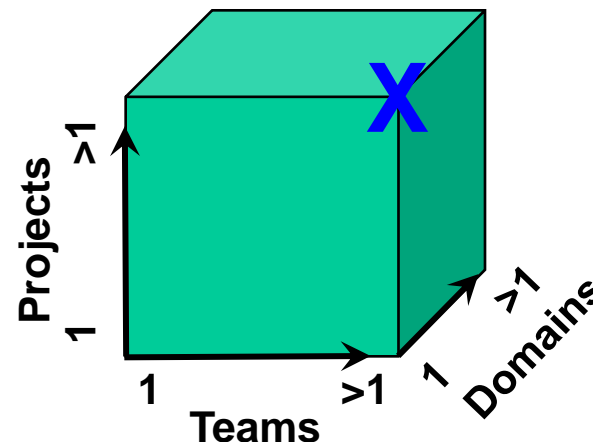
MODELING

- Defect detection techniques: Disciplined team-based peer reviews

Enables?

- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

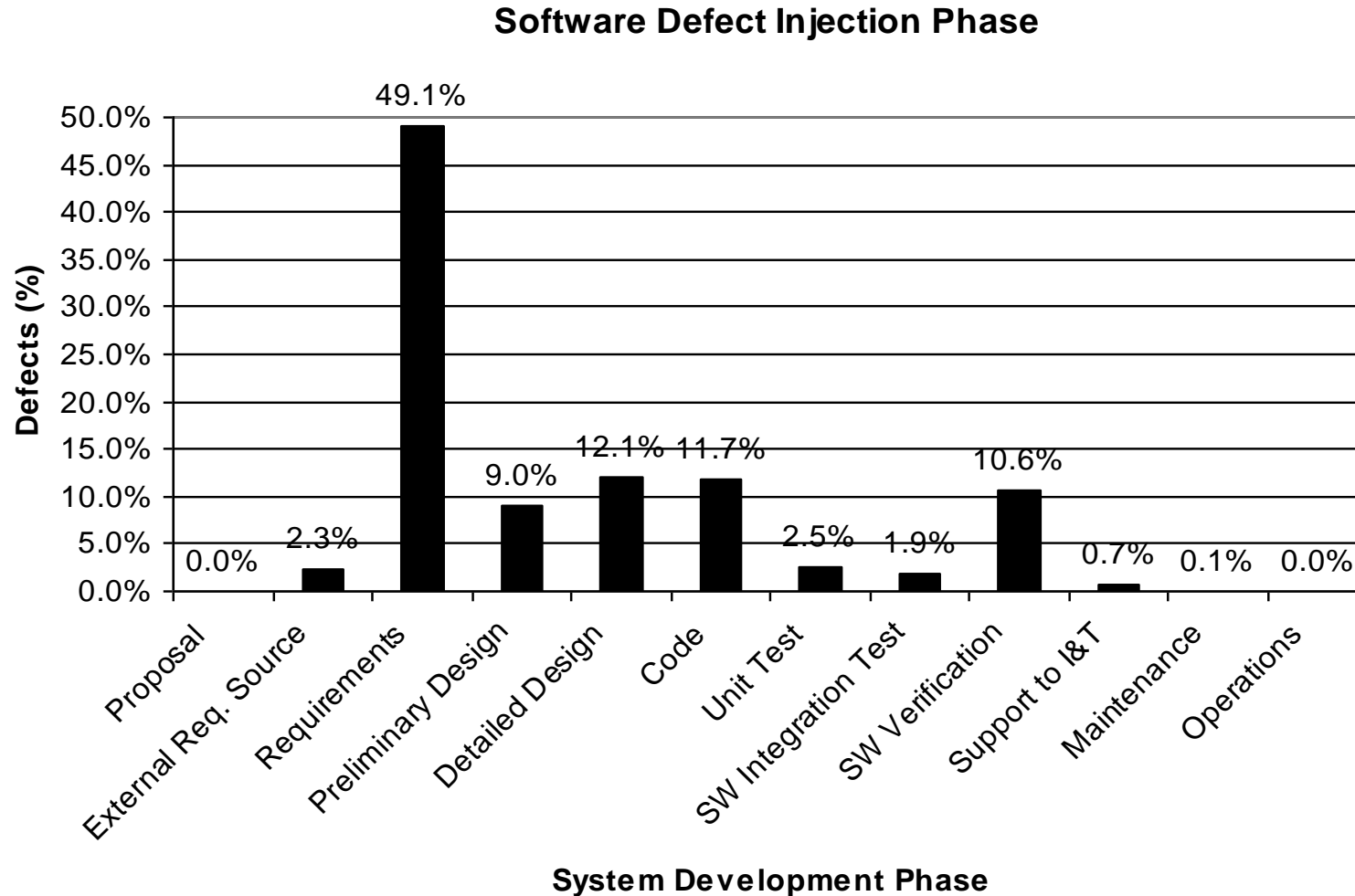
SCALING DIMENSIONS



BENEFITS and TRADEOFFS

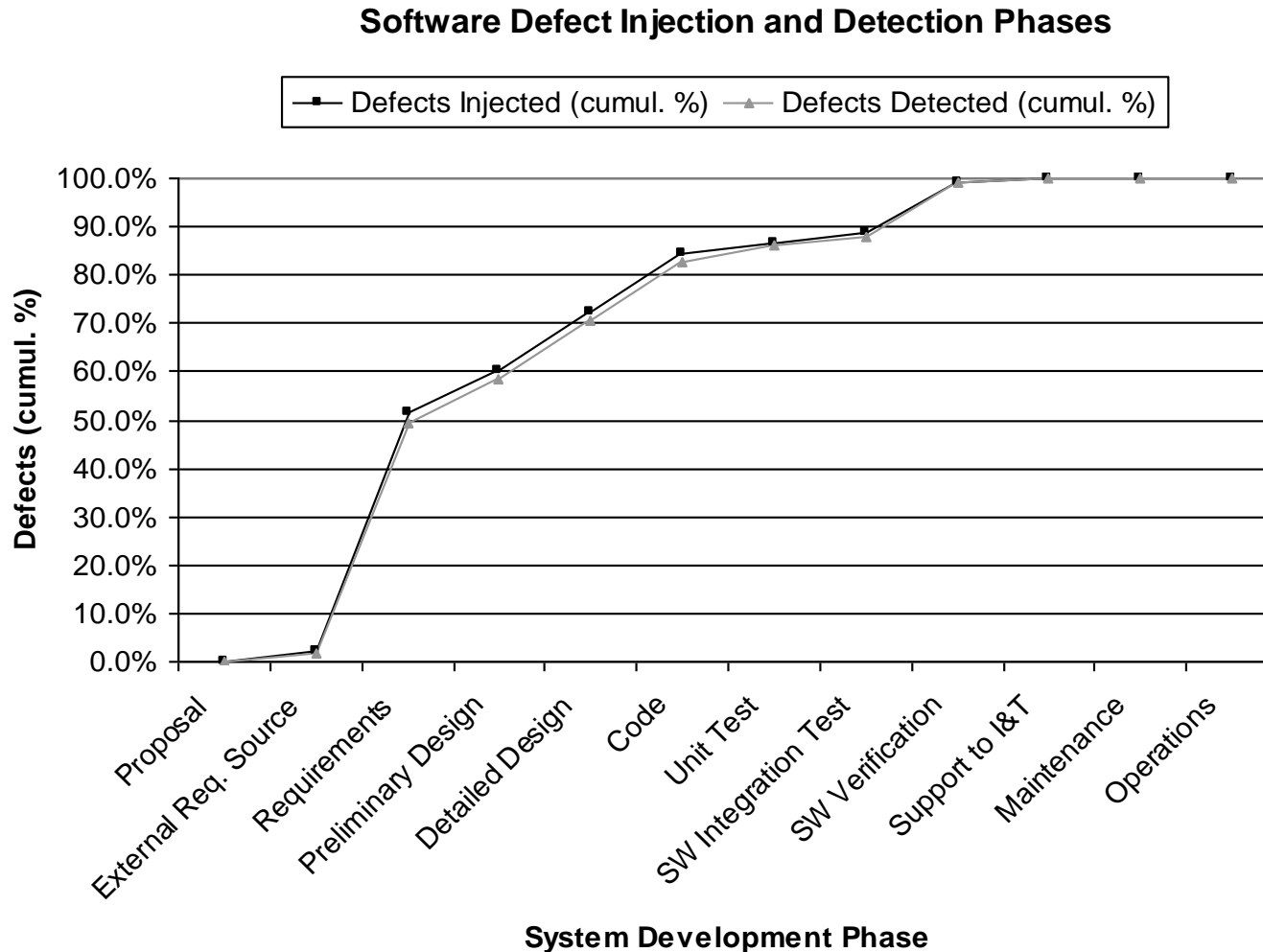
- Early lifecycle defect detection
- Low out-of-phase defect rates
- High return-on-investment for prevention

Analyses of Software Defect Injection Phases Reveals Distributions



- **Distribution of software defect injection phases based on using peer reviews across 12 system development phases**
- **3418 defects, 731 peer reviews, 14 systems, 2.67 years**
- **49% of defects injected during requirements phase**

Analyses of Software Defect Injection and Detection Phases Reveal Distributions and Gaps



- Cumulative distribution of software defect injection and detection phases based on using peer reviews across 12 system development phases
- 3418 defects, 731 peer reviews, 14 systems, 2.67 years
- 50% defects injected by requirements, 70% by detailed design; Gap shows leakage

Web-Based Workflow Tools and Infrastructure Support Software Process Flow

Embedded Software Processes Flow Guide (PRC) - Microsoft Internet Explorer provided by Northrop Grumman Corporation

Requirements Development and Management Phase

- Tasks to be Completed Before IDR#1: Develop concept of operations, Evaluate concept of operations impacts, Develop glossary and acronym list, Establish stakeholder requirements, Create requirements specification, Establish performance requirements, Define acceptance method, Identify External Interfaces, Identify component interfaces, Define verification method, Perform risk analysis, Define TMs, Validate stakeholder requirements, Define builds based on stakeholder needs, Allocate requirements to builds, Create traceability matrices, Stakeholder Requirements Peer Reviews, Obtain stakeholder approval, Control requirements changes, Generate IDR#1 Review Package.

Preliminary Design Phase

- Tasks to be Completed Before IDR#2: Design methods established, Design tool identification, Create SW Test Plan, Product component make/buy/use plan, Alternative product architecture criteria, Architecture established, Detailed alternate architecture solutions/selection criteria, Architecture trade studies/risk analyses, Architecture selection/expansion, Architecture decision rationale, Feasibility analysis, Prototype build (if necessary), Updated concept of operations, Product/component requirements updated/evaluated, Updated SW artifacts based on solution, Design/Update external and internal I/Fs, Traceability matrices, Establish interface repository, Incremental requirements allocation (as needed), Establish version control (as needed), Updated reason plans (as needed), Peer review with stakeholders, Verify product components & I/F designs, Control approved design work products, Update TMs, Generate IDR#2 Review Package.

Detailed Design Phase

- Tasks to be Completed Before IDR#3: Product/component/unit I/F design criteria, Product/component/unit evaluation, Updated SW artifacts based on solution, Develop product component/unit definitions, Component/unit requirements allocation, Traceability matrices, Design architecture, Design external & internal I/F descriptions, Updated interface descriptions.

Implementation & Integration Phase

- Tasks to be Completed Before IDR#4: Define coding standards, Define user interface, Define end user documentation, Define file header format, Implement code unit/subcomponents, Peer review code unit/subcomponents, Perform unit tests, Peer review unit tests, Configuration management of code unit/subcomponents, Peer review documentation, Configuration managed documents, Update traceability matrices, Update technical data package, Maintain Software Development Folders.

Verification & Validation Phase

- Tasks to be Completed Before IDR#5: Control/Maintain SW Test Plan, Control/Maintain SW Test Procedures, Control/Maintain Test Verification Matrix, Peer review test results, Acquire V&V test tools, Perform V&V per SW Test Plan, Document Discrepancies in SDRs, Maintain Test Development Folders, Document Results in SW Test Report, Update TPLs, Generate IDR#5 Review Package.

Note: This summary chart provided for information only. The SPRM (MNL370) and SSPM (MNL377) are the controlling documents.

** For an DR review, achievement of an evaluation score of ≥ 2.5 authorizes progression to the next stage.

Process flow

SEPG Peer Review - Microsoft Internet Explorer provided by Northrop Grumman Corporation

SEPG Home | Peer Reviews | Action Items | Create New | Metrics | FAQ | Help | Admin

Peer Reviews Search

Program - Subproject: All

Search For: Title: All

Filter By: Author: Abe, Gerald; Aguilu, Lemuel; Allen, Francis; Almeida, Rudy

Filter By: Actual Date: []

Sort By: Column: PR Number | Ascending | Descending

Peer Review Author	Actual Date	Date Submitted	Opened Action Items
Amy Yu	7/28/2004	None	11
Amy Yu	7/26/2004	None	7
Chris Chikami	7/23/2004	7/26/2004	3
Eric Liang	7/23/2004	None	0

Total Record Count: 317

Peer reviews

SEPG Peer Review - Microsoft Internet Explorer provided by Northrop Grumman Corporation

Program: SIM Subproject: Flight

Prepared By (Peer Review Author): []

Software Project Management/Team Lead: [] Approve

Peer Review Exit Criteria (To be completed by the Peer Review meeting)

- Are the action items, key comments, agreements, and meeting properly entered into the peer review portion of the SPRM?
- Are all action item assignees aware and knowledgeable of the actions assigned to them?*
- Do all action item assignees know to status their action items on the peer review portion of the SPRM metrics spreadsheet?*
- Are all peer review work products captured in the project data repositories (e.g., software development folder)?*
- Were the objectives of the peer review met?*
- Was the review material completely reviewed?*
- Does the product require extensive rework that warrants a re-review?*
- Based on checklist steps 5, 6, and 7, is a re-review of the peer review material necessary?*

Local intranet

Exit checklist

SEPG Peer Review - Microsoft Internet Explorer provided by Northrop Grumman Corporation

Peer Reviews Search Results

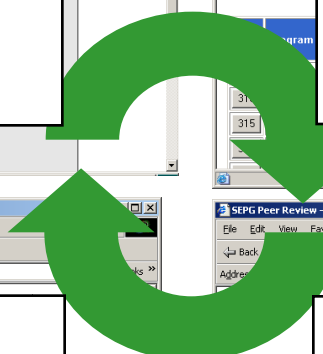
Assignee: All; Abe, Gerald; Aguilu, Lemuel; Allen, Francis; Almeida, Rudy

Search | Reset | Download To Excel

PR # - AI #	Program - Subproject	Peer Review Title	Action Item Description	Status	Originator	Assignee	Due Date
317-1	NPOESS - Flight Software - FSW	Exec Processor Selftest Top Level Design	GET global data types from the whole team.	New Assigned	Kevin Guthrie	Amy Yu	8/10/2004
317-2	NPOESS - Flight Software - FSW	Exec Processor Selftest Top Level Design	ram_checksum, fault_enable's description should match requirement and it's range should be late/not late.	New Assigned	Marcel LeRutte	Amy Yu	8/11/2004
317-3	NPOESS - Flight Software - FSW	Exec Processor Selftest Top Level Design	Check all data types against requirements.	New Assigned	Marcel LeRutte	Amy Yu	8/11/2004

Total Record Count: 2287

Action items



Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

PRINCIPLES

SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters

ANALYSIS

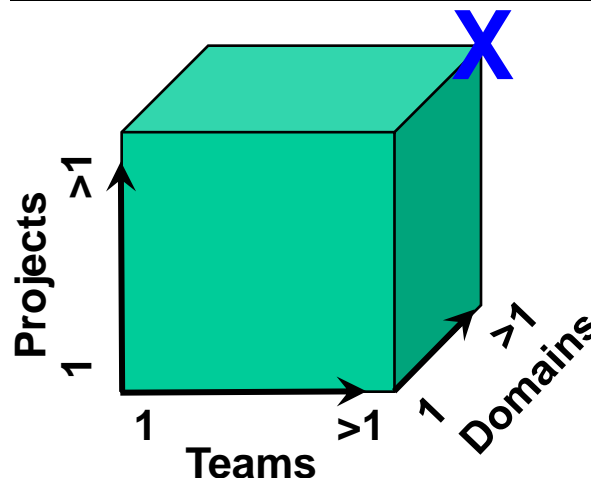
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis

MODELING

- Defect detection techniques: Disciplined team-based peer reviews
- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

Enables?

SCALING DIMENSIONS

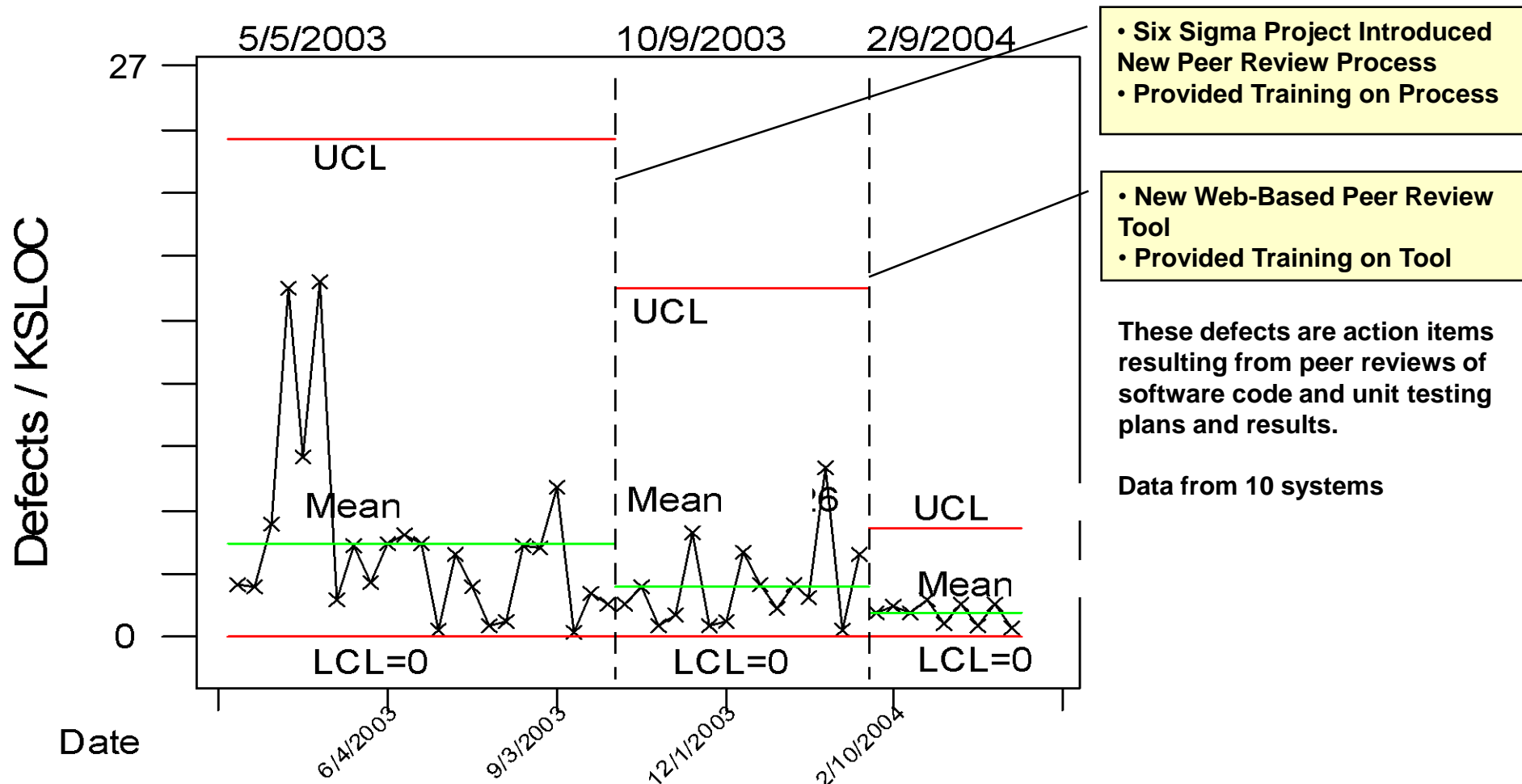


BENEFITS and TRADEOFFS

- Early identification of high defect or high effort components
- Statistical process control
- Pro-active process guidance

Data-Driven Statistical Analyses Identify Trends, Outliers, and Process Improvements for Defects

Defect Density for Code / Unit Test Peer Reviews

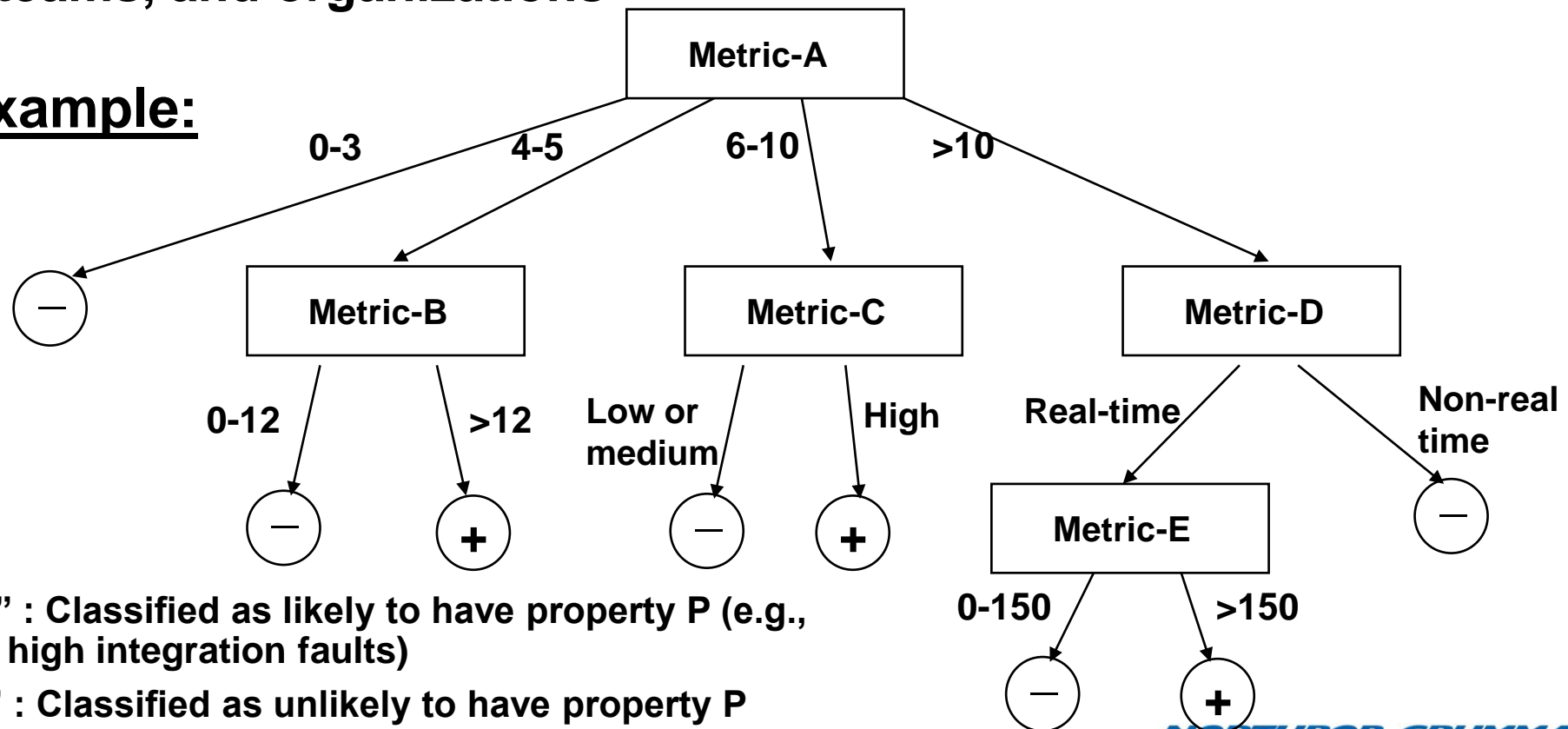


- Control chart of metric data from example Six Sigma projects focusing on fault (or defect) density in peer reviews of software components
- Process improvements decreased variances and decreased means

Measurement-Driven Decision Models (Trees, Networks) Predict High-Risk Software Components

- Focus on high-payoff areas: the 80:20 rule
- Generate decision trees or networks automatically
 - Scalable to large systems
 - Leverage previous experience and calibrate to new environments
- Integrate measurements from processes, products, projects, teams, and organizations

Example:

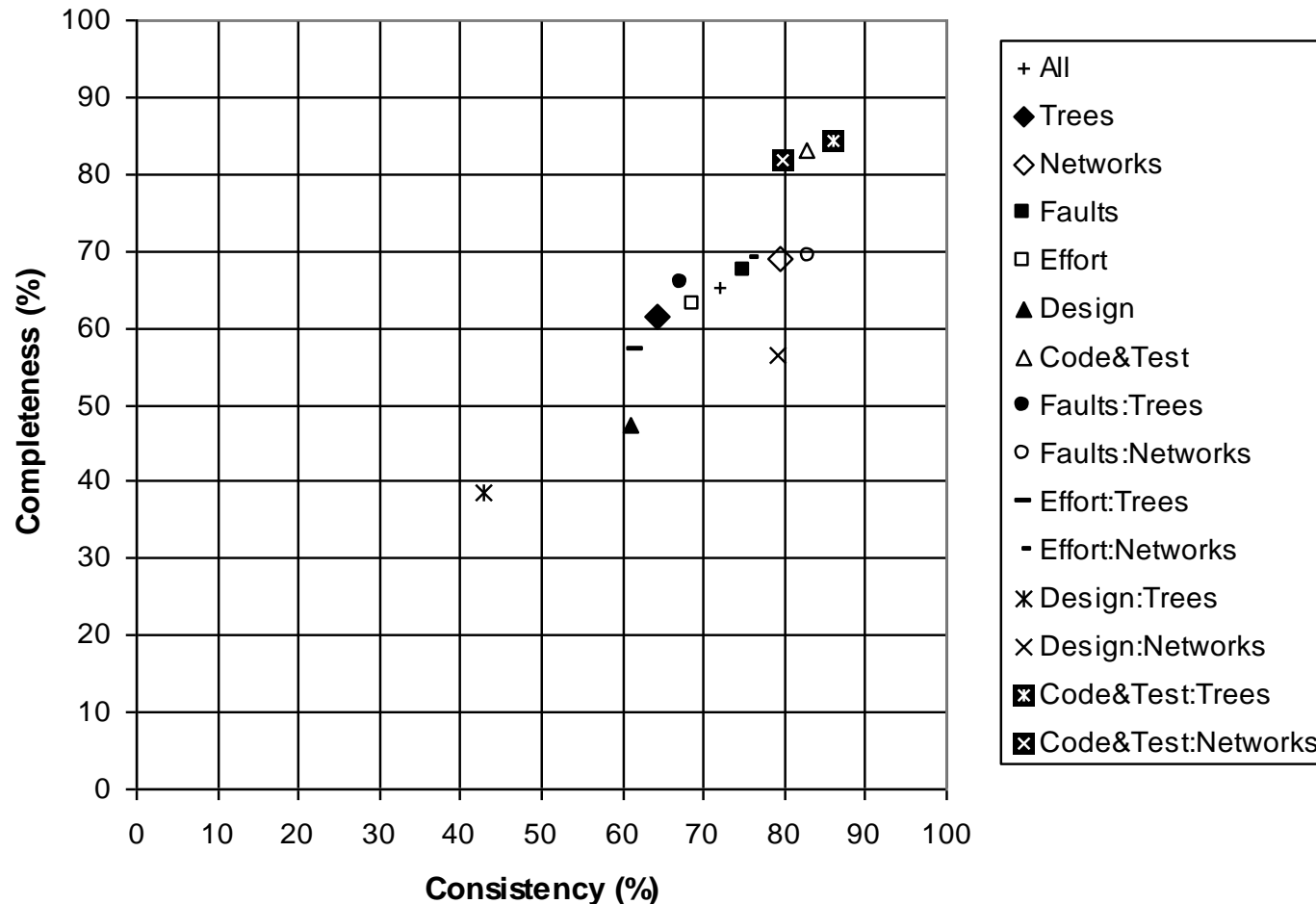


“+” : Classified as likely to have property P (e.g., high integration faults)

“-” : Classified as unlikely to have property P

Predictive Models Identify Leading Indicators of High-Fault and High-Effort Components

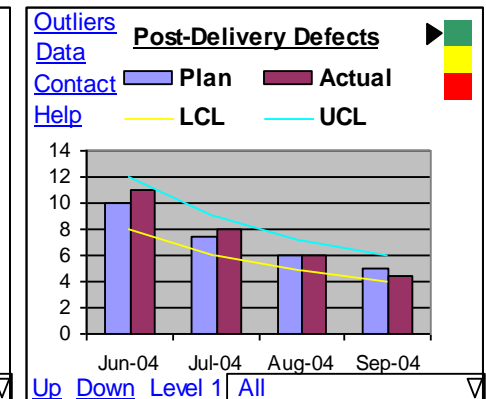
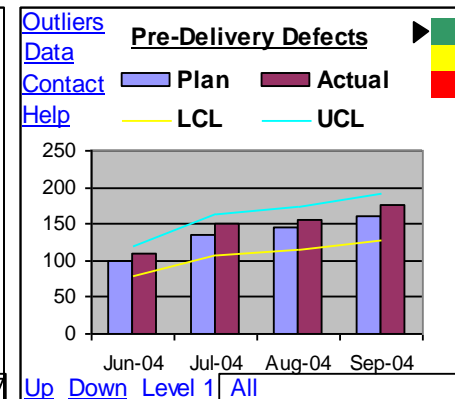
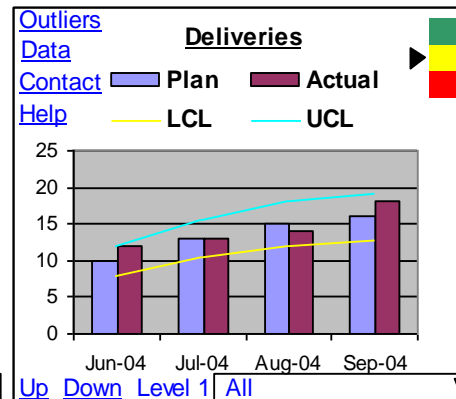
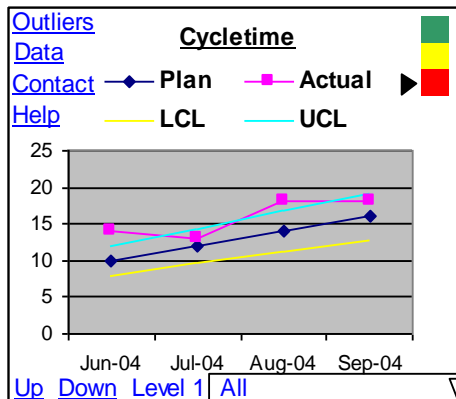
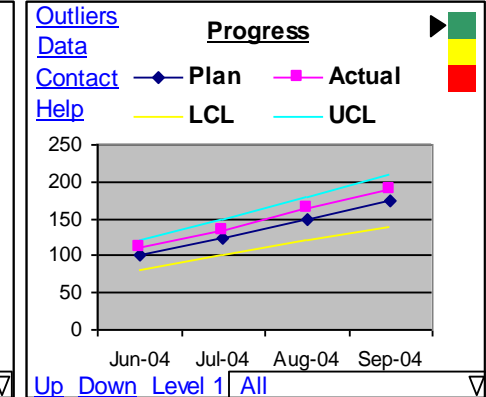
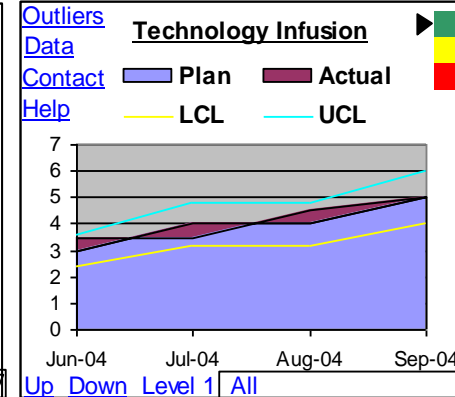
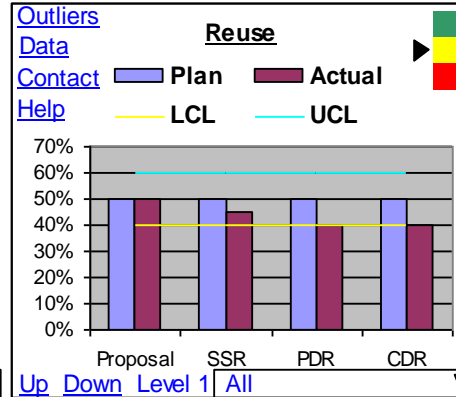
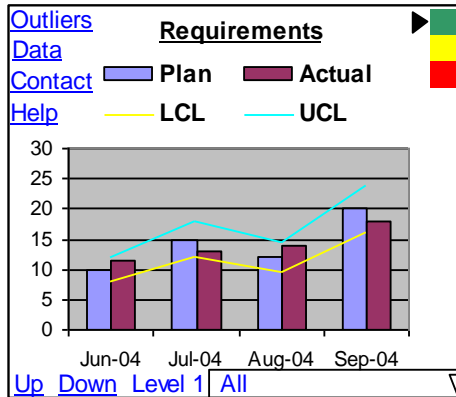
Model Accuracies and Tradeoffs



- Data from 16 NASA systems. 1920 model variations.
- Consistency is 100% minus percent false positives. Completeness is 100% minus percent false negatives.

Interactive Metric Dashboards Provide Framework for Visibility, Flexibility, Integration, Automation

DASHBOARD Metrics: Organization: Project: Manager: Contact: Status:



- Interactive metric dashboards incorporate a variety of information and features to help developers and managers characterize progress, identify outliers, compare alternatives, evaluate risks, and predict outcomes

Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

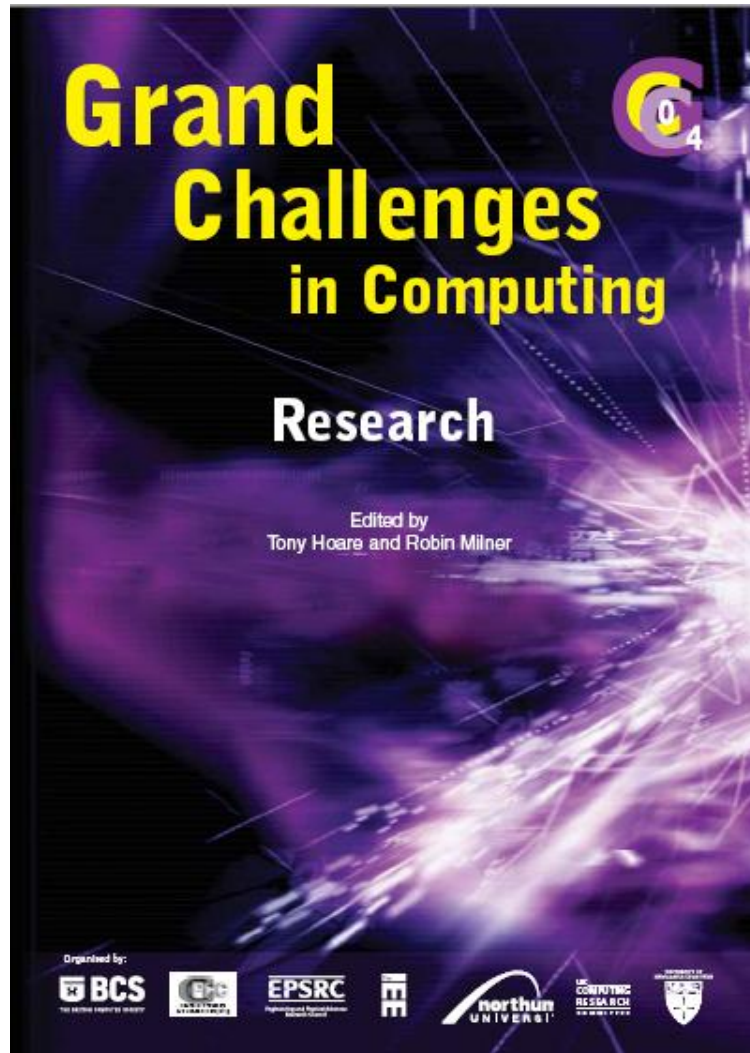
- **Systems and software engineering strategies, principles, benefits, and tradeoffs**
- **Example large-scale mission-critical embedded software system**
- **Investigations of synthesis, analysis, and modeling principles**
 - Synthesis: Lifecycle models
 - Synthesis: System architectures
 - Analysis: Reuse analysis
 - Analysis: Structure analysis
 - Modeling: Defect detection techniques
 - Modeling: Measurement and prediction



- **Conclusions and future work**

Define "Grand Challenges" Problems for Systems and Software Engineering

Example from Computing (2004)



Contents

Introduction

Tony Hoare and Robin Milner

GC1

In Vivo-in Silico (ViS): the virtual worm, weed and bug
Ronan Sleep

GC2

Science for global ubiquitous computing
Marta Kwiatkowska and Vladimiro Sassone

GC3

Memories for life: managing information over a human lifetime
Andrew Fitzgibbon and Ehud Reiter

GC4

Scalable ubiquitous computing systems
Jon Crowcroft

GC5

The architecture of brain and mind
Aaron Sloman

GC6

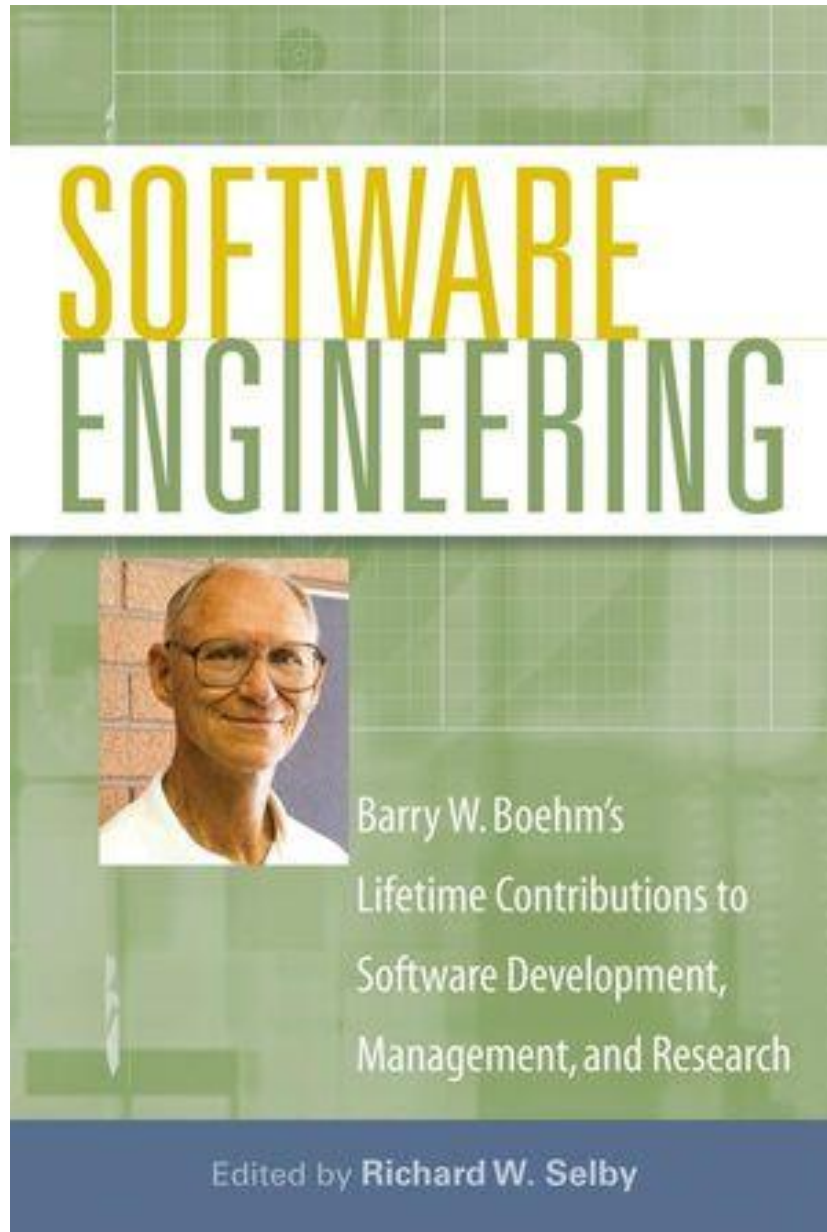
Dependable systems evolution
Jim Woodcock

GC7

Journeys in non-classical computation
Susan Stepney

Source: <http://www.ukcrc.org.uk/gcresearch.pdf>

Software Engineering Book Captures Best Practices for Economics, Quality, Process, Risk Management



- **Richard W. Selby, Editor, Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research, IEEE Computer Society and John Wiley & Sons: New York, May 2007, ISBN 9780-4701-48730.**

- **Rick.Selby@NGC.com**