Interactive Models for Display, Drilldown, and Decision-Making on Large-Scale Systems

April 2009

Rick Selby
Director of Software Products
Northrop Grumman Aerospace Systems
310-813-5570, Rick.Selby@NGC.com

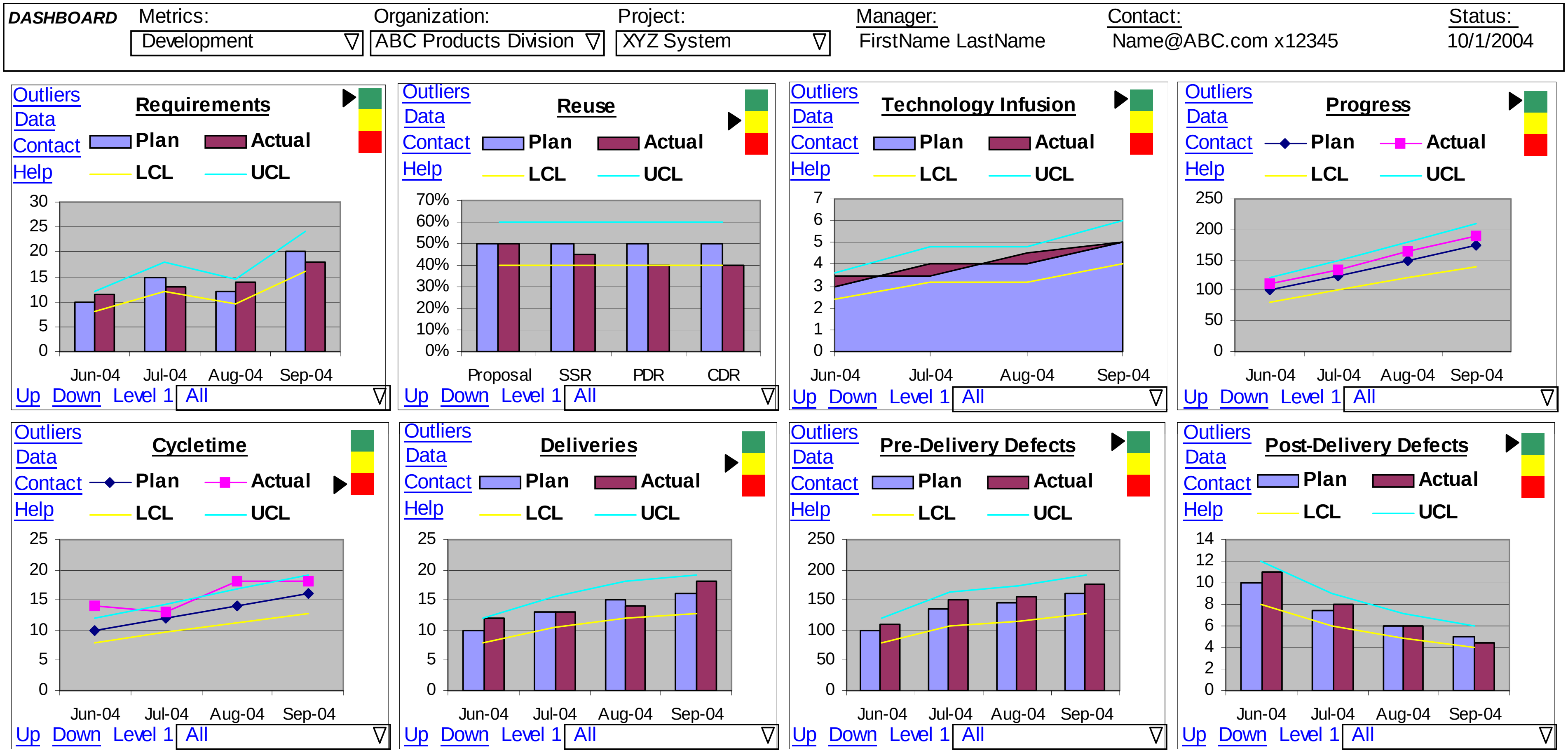
Adjunct Professor of Computer Science
University of Southern California

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**

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



DASHBOARD Metrics: Development ▽ Organization: ABC Products Division ▽ Project: XYZ System ▽ Manager: FirstName LastName Contact: Name@ABC.com x12345 Status: 10/1/2004

- **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 Engineering Principles, Benefits, and Tradeoffs

	PRINCIPLES		BENEFITS and TRADEOFFS
SYNTHESIS	Lifecycle models: Frequent synchronized design cycles and system releases	Enables?	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	Enables?	User-customizability Multi-platform portability Automated testing
ANALYSIS	Reuse analysis: Reconfigurable component-driven development	Enables?	Sustainable multi-project reuse Lower component defect rates Lower component development effort
	Structure analysis: Inter-component connectivity analysis	Enables?	Lower component defect rates Lower component defect correction effort Lower component development effort
MODELING	Defect detection techniques: Disciplined team-based peer reviews	Enables?	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	Enables?	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

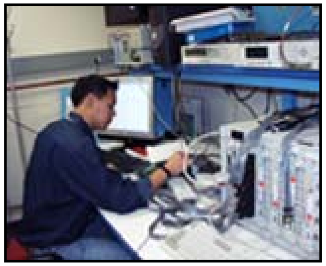
Overview

- **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

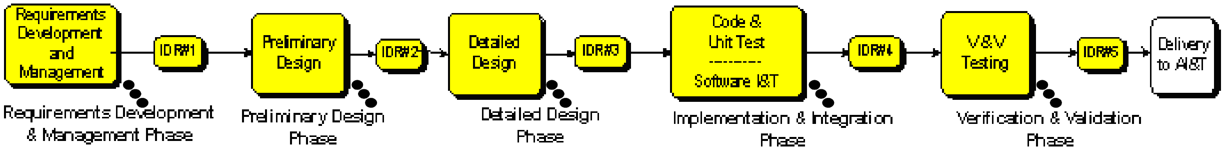
- **Embedded software for**
  - **Advanced robotic spacecraft platforms**
  - **High-bandwidth satellite payloads**
  - **High-power laser systems**
- **Emphasis on both system management and payload software**
- **Reusable, reconfigurable software architectures and components**
- **Languages: O-O to 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


Requirements Development and Management

IDR#1

Preliminary Design

IDR#2

Detailed Design

IDR#3

Code & Unit Test

Software I&T

IDR#4

V&V Testing

IDR#5

Delivery to AI&T

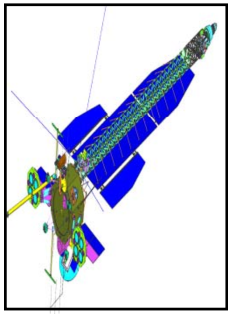
Requirements Development & Management Phase

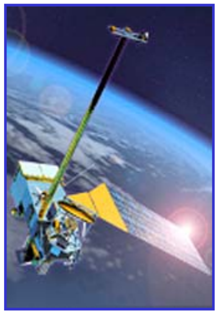
Preliminary Design Phase

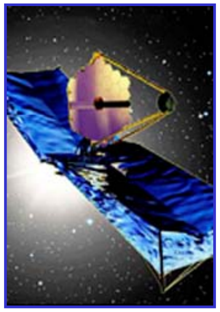
Detailed Design Phase

Implementation & Integration Phase

Verification & Validation Phase


Prometheus / JIMO


NPOESS


JWST


EOS Aqua/Aura


Chandra


GeoLITE

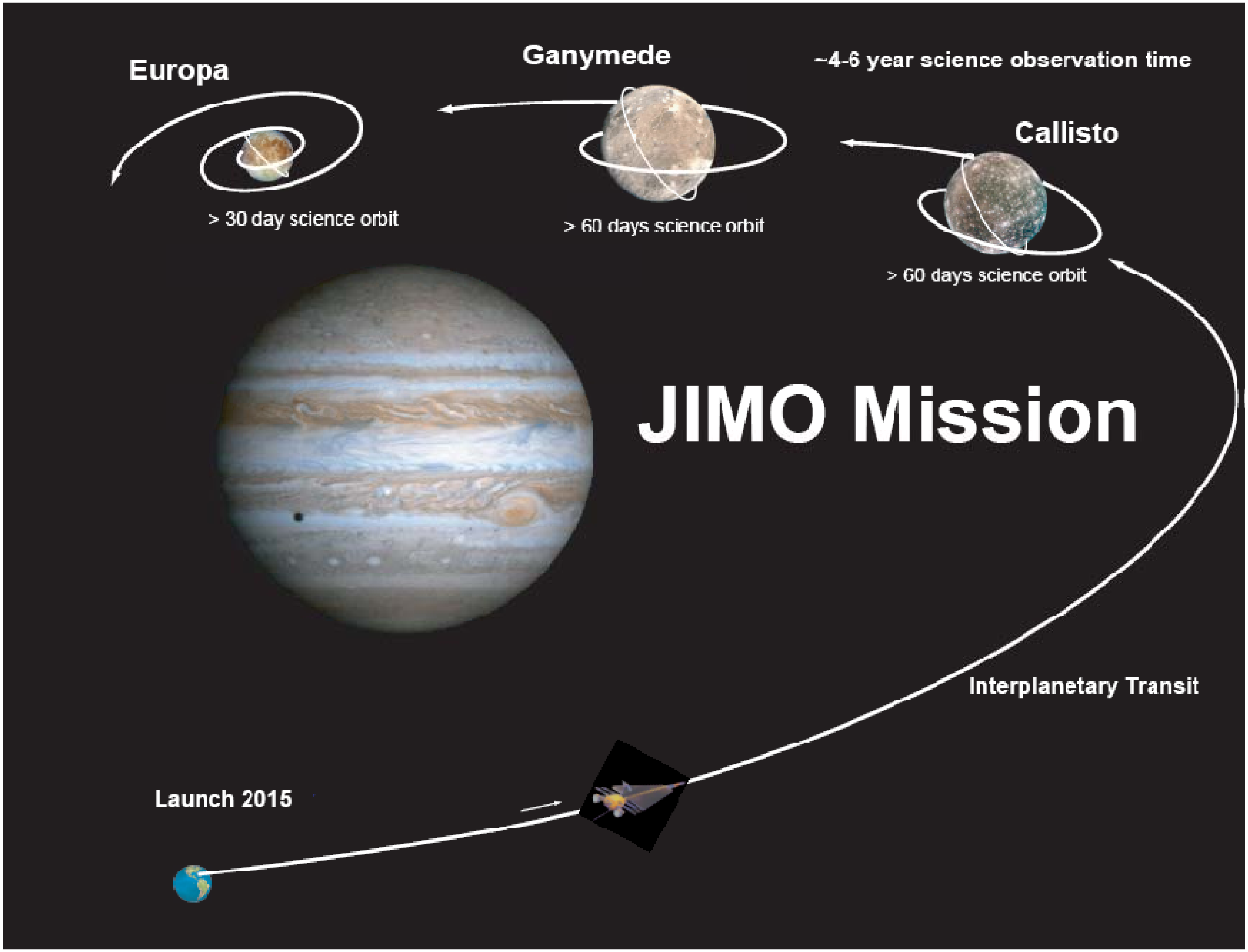

AEHF


MTHEL


Airborne Laser

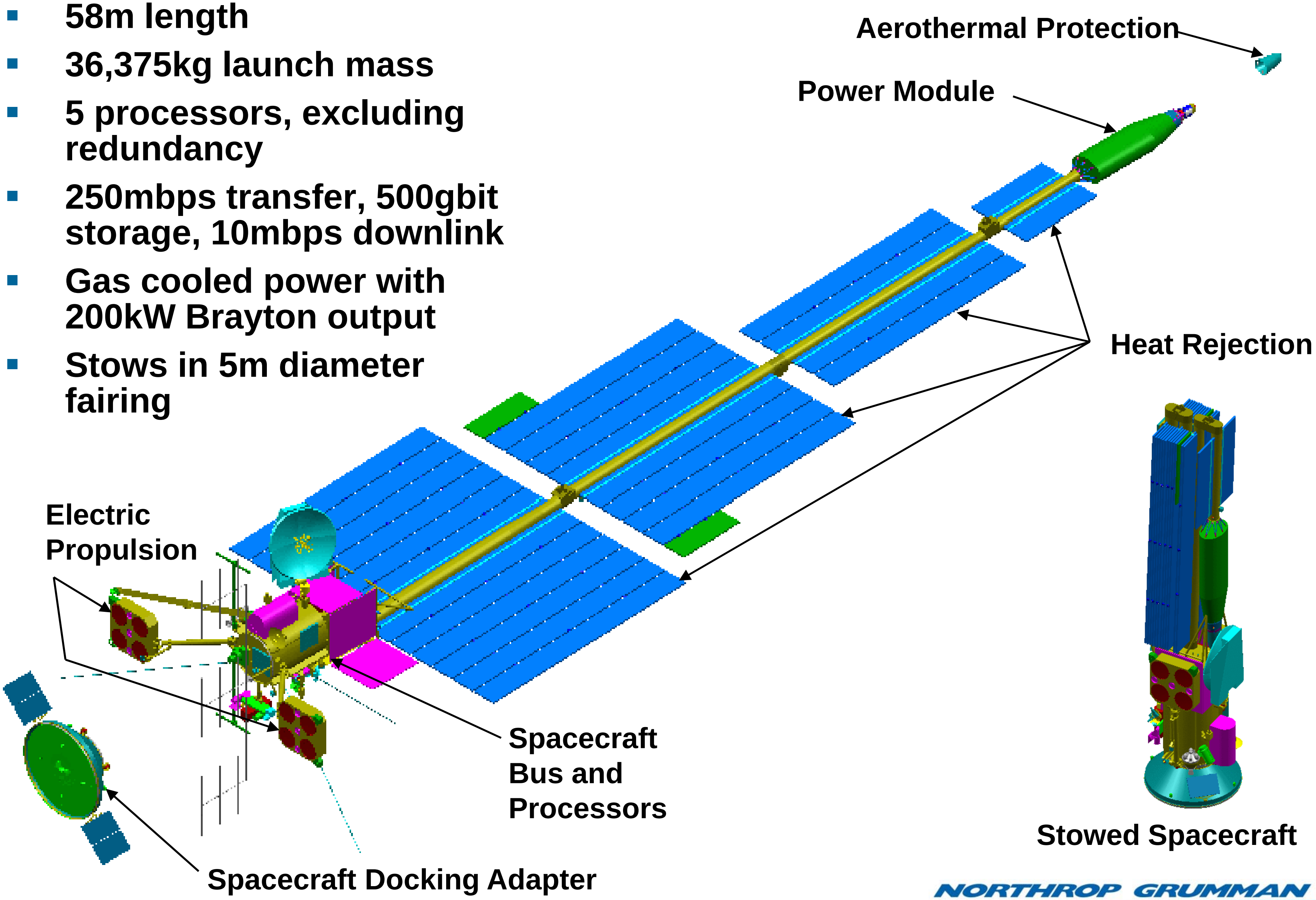

Restricted

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



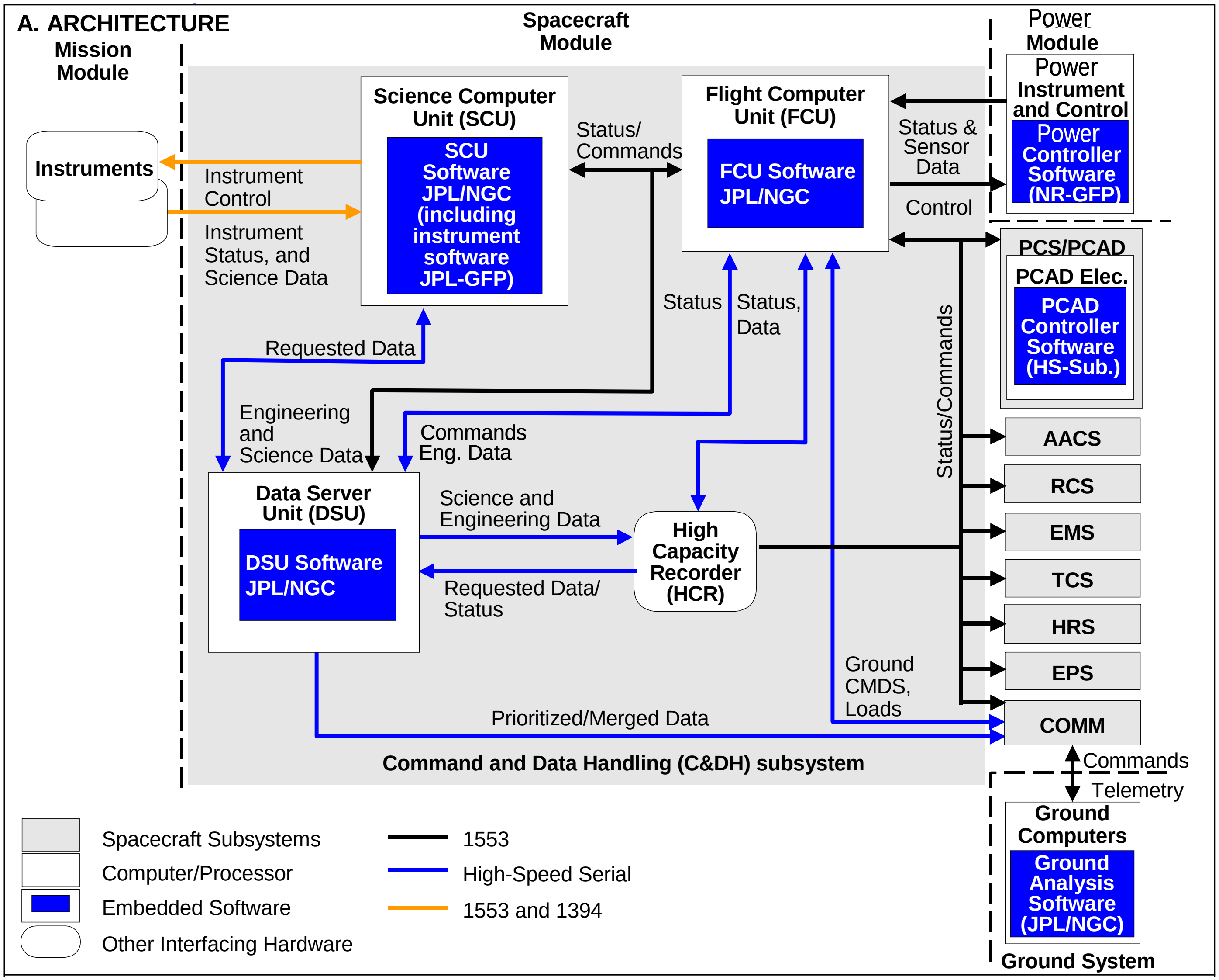
Prometheus Spacecraft for JIMO and Related Missions Enables Data-Intensive Science

- 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

SYNTHESIS

- Lifecycle models: Frequent synchronized design cycles and system releases

Enables?

BENEFITS and TRADEOFFS

- 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

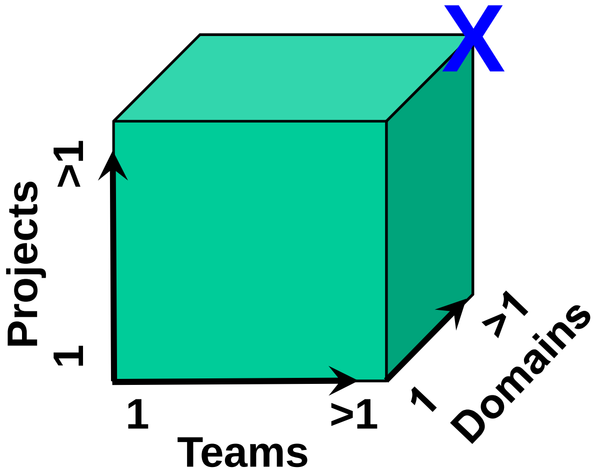
ANALYSIS

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

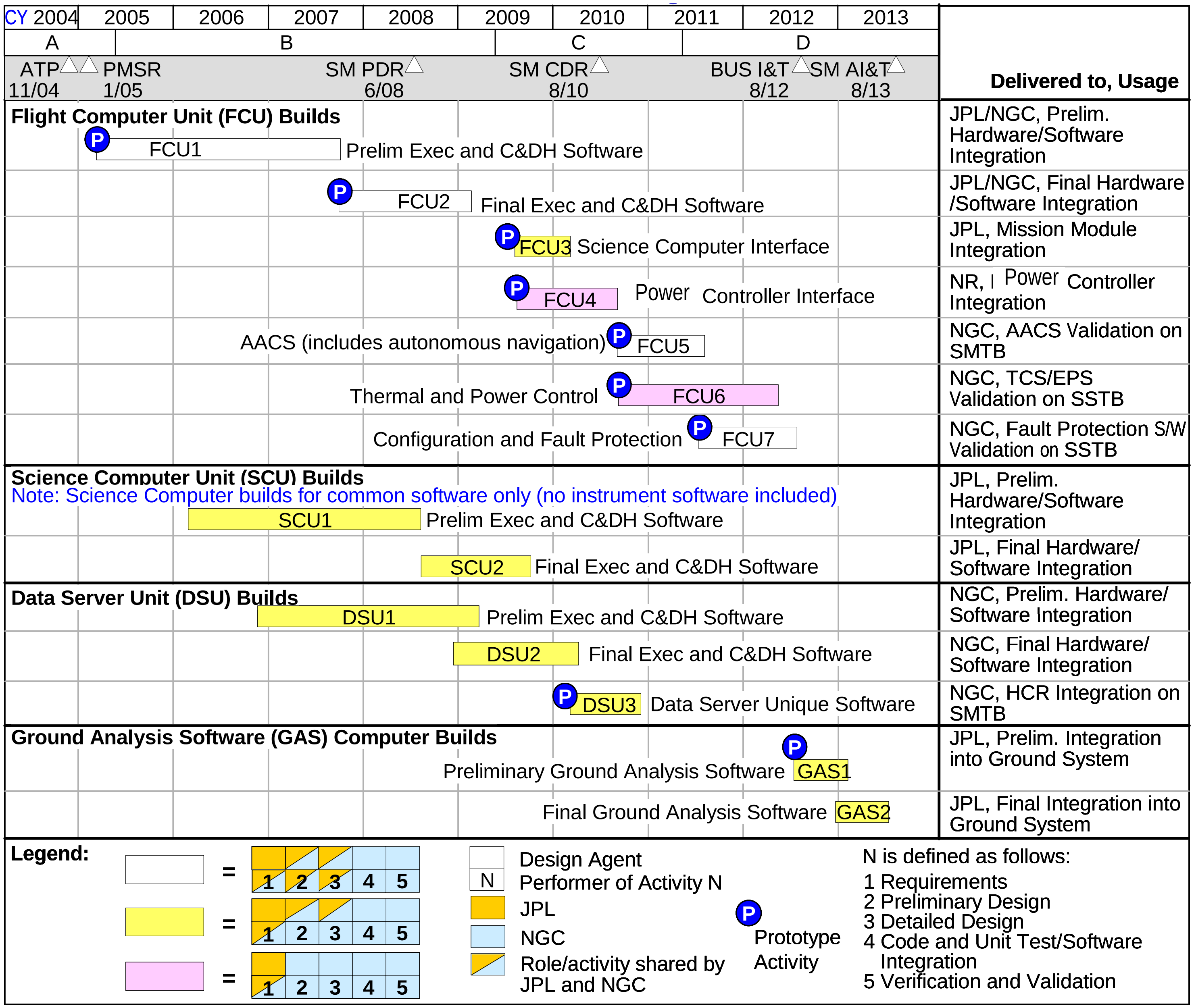
MODELING

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

SCALING DIMENSIONS



Incremental Software Builds Deliver Early Capabilities and Accelerate Integration and Test



		Delivered to, Usage
Flight Computer Unit (FCU) Builds		
FCU1	Prelim Exec and C&DH Software	JPL/NGC, Prelim. Hardware/Software Integration
FCU2	Final Exec and C&DH Software	JPL/NGC, Final Hardware /Software Integration
FCU3	Science Computer Interface	JPL, Mission Module Integration
FCU4	Power Controller Interface	NR, I Power Controller Integration
FCU5	AACS (includes autonomous navigation)	NGC, AACS Validation on SMTB
FCU6	Thermal and Power Control	NGC, TCS/EPS Validation on SSTB
FCU7	Configuration and Fault Protection	NGC, Fault Protection S/W Validation on SSTB
Science Computer Unit (SCU) Builds Note: Science Computer builds for common software only (no instrument software included)		
SCU1	Prelim Exec and C&DH Software	JPL, Prelim. Hardware/Software Integration
SCU2	Final Exec and C&DH Software	JPL, Final Hardware/ Software Integration
Data Server Unit (DSU) Builds		
DSU1	Prelim Exec and C&DH Software	NGC, Prelim. Hardware/ Software Integration
DSU2	Final Exec and C&DH Software	NGC, Final Hardware/ Software Integration
DSU3	Data Server Unique Software	NGC, HCR Integration on SMTB
Ground Analysis Software (GAS) Computer Builds		
GAS1	Preliminary Ground Analysis Software	JPL, Prelim. Integration into Ground System
GAS2	Final Ground Analysis Software	JPL, Final Integration into Ground System

Legend:

☐ = 1 2 3 4 5
🟨 = 1 2 3 4 5
🟪 = 1 2 3 4 5

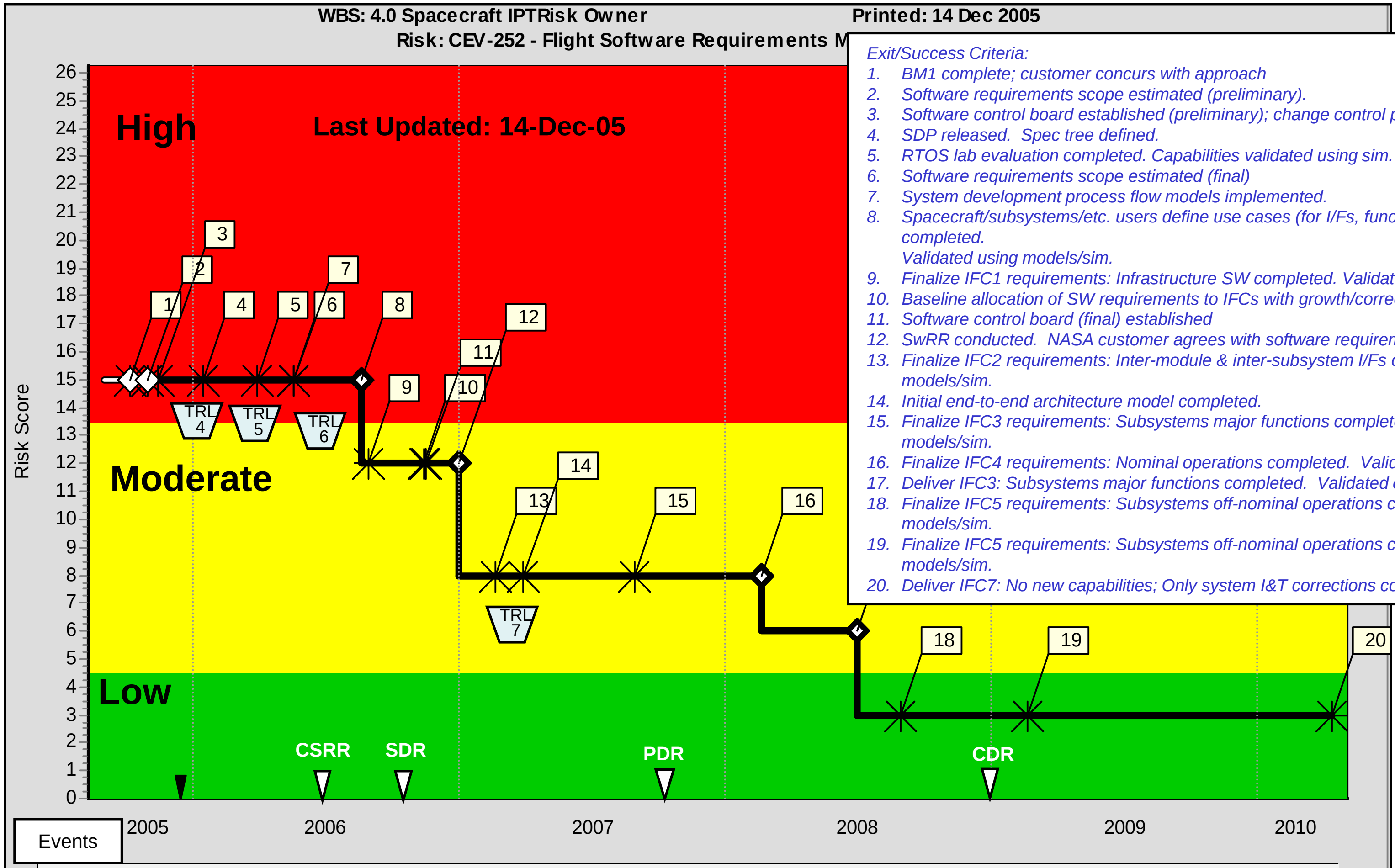
☐ Design Agent
N Performer of Activity N
🟧 JPL
🟦 NGC
◩ Role/activity shared by JPL and NGC

P Prototype Activity

N is defined as follows:
1 Requirements
2 Preliminary Design
3 Detailed Design
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



Exit/Success Criteria:

1. *BM1 complete; customer concurs with approach*
2. *Software requirements scope estimated (preliminary).*
3. *Software control board established (preliminary); change control process established.*
4. *SDP released. Spec tree defined.*
5. *RTOS lab evaluation completed. Capabilities validated using sim.*
6. *Software requirements scope estimated (final)*
7. *System development process flow models implemented.*
8. *Spacecraft/subsystems/etc. users define use cases (for I/Fs, functions, nominal ops, off-nominal ops, etc.) completed.*
   Validated using models/sim.
9. *Finalize IFC1 requirements: Infrastructure SW completed. Validated requirements using models/sim.*
10. *Baseline allocation of SW requirements to IFCs with growth/correction/deficiency completed.*
11. *Software control board (final) established*
12. *SwRR conducted. NASA customer agrees with software requirements.*
13. *Finalize IFC2 requirements: Inter-module & inter-subsystem I/Fs completed. Validated requirements using models/sim.*
14. *Initial end-to-end architecture model completed.*
15. *Finalize IFC3 requirements: Subsystems major functions completed. Validated requirements using models/sim.*
16. *Finalize IFC4 requirements: Nominal operations completed. Validated requirements using models/sim.*
17. *Deliver IFC3: Subsystems major functions completed. Validated capabilities using sim.*
18. *Finalize IFC5 requirements: Subsystems off-nominal operations completed. Validated requirements using models/sim.*
19. *Finalize IFC5 requirements: Subsystems off-nominal operations completed. Validated requirements using models/sim.*
20. *Deliver IFC7: No new capabilities; Only system I&T corrections completed. SW complete for 1st mission.*

1 : Conduct BM1
2 : Estimate software requirements scope (preliminary)
3 : Establish software control board (preliminary)
4 : Release SDP (with spec tree, change process)
5 : Complete RTOS lab evaluation; Validate capabilities using sim
6 : Estimate software requirements scope (final)
7 : Implement system development process flow models
8 : Spacecraft/etc users define/val/sim use-cases (I/F, funct, off-nom, ...)
9 : Finalize/val/sim IFC1 requirements: Infrastructure SW
10 : Baseline allocation of SW requirements to IFCs with growth/correction
11 : Establish software control board (final)
12 : Conduct SwRR
13 : Finalize/val/sim IFC2 requirements: Inter-module & inter-subsystem I/F
14 : Initial end-to-end architecture model
15 : Finalize/val/sim IFC3 requirements: Subsystems major functions
16 : Finalize/val/sim IFC4 requirements: Nominal operations
17 : Deliver IFC3: Subsystems major functions
18 : Finalize/val/sim IFC5 requirements: Subsystems off-nominal operations
19 : Finalize/val/sim IFC6 requirements: System off-nominal operations
20 : Deliver IFC7: No new capabilities; Only corrections; 1st mission ready

━ Planned Risk Level	◆ Planned (Solid=Linked, Hollow=Unlinked)	✳ Control Points
Actual Risk Level	Completed	◇ Completed

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

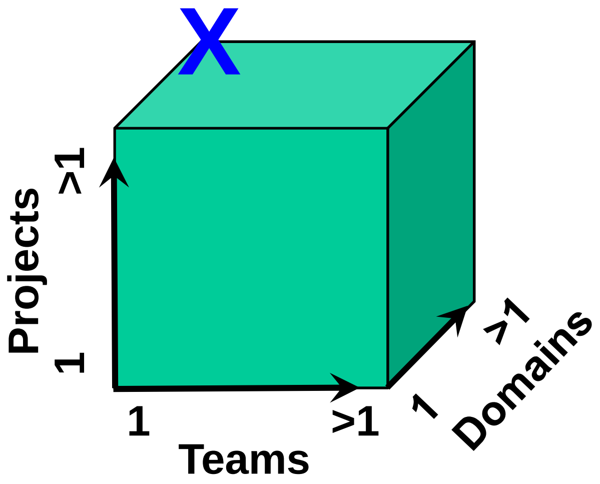
ANALYSIS

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

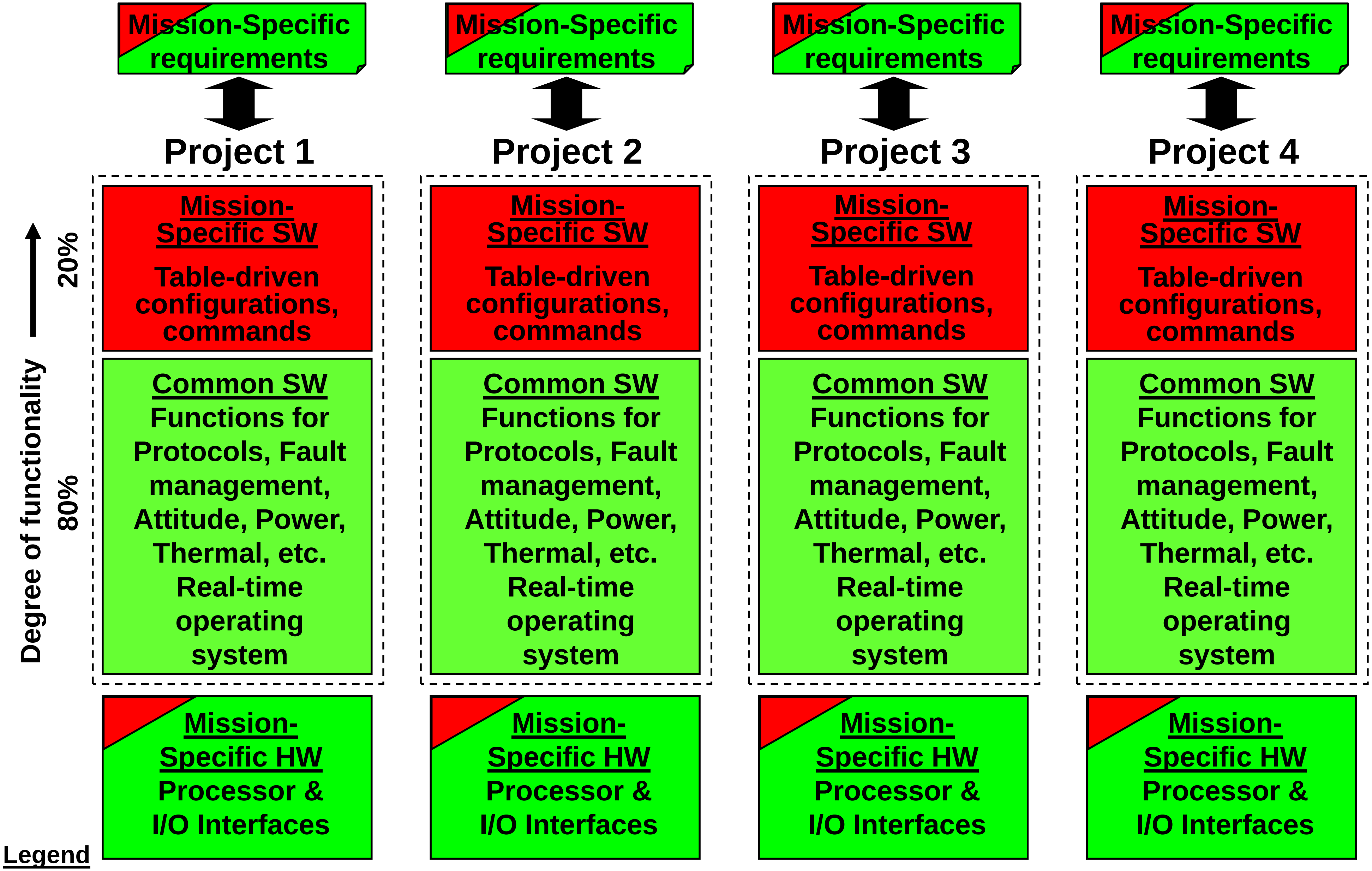
MODELING

- 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



Degree of functionality

20%

80%

Mission-Specific requirements

Project 1

Mission-Specific SW

Table-driven configurations, commands

Common SW

Functions for Protocols, Fault management, Attitude, Power, Thermal, etc.

Real-time operating system

Mission-Specific HW

Processor & I/O Interfaces

Mission-Specific requirements

Project 2

Mission-Specific SW

Table-driven configurations, commands

Common SW

Functions for Protocols, Fault management, Attitude, Power, Thermal, etc.

Real-time operating system

Mission-Specific HW

Processor & I/O Interfaces

Mission-Specific requirements

Project 3

Mission-Specific SW

Table-driven configurations, commands

Common SW

Functions for Protocols, Fault management, Attitude, Power, Thermal, etc.

Real-time operating system

Mission-Specific HW

Processor & I/O Interfaces

Mission-Specific requirements

Project 4

Mission-Specific SW

Table-driven configurations, commands

Common SW

Functions for Protocols, Fault management, Attitude, Power, Thermal, etc.

Real-time operating system

Mission-Specific HW

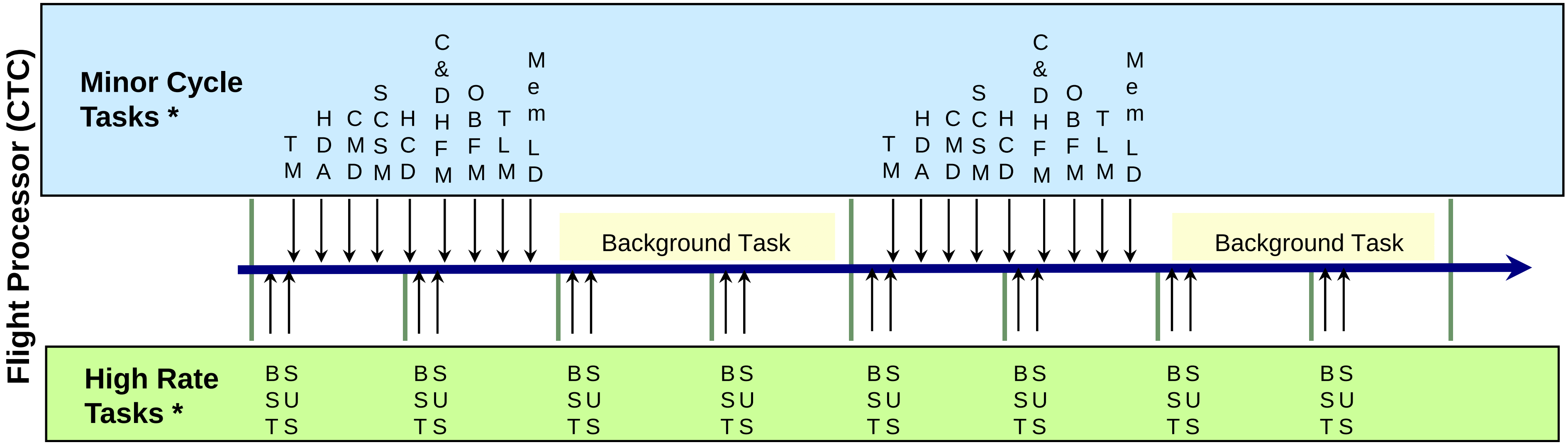
Processor & I/O Interfaces

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**

Enables?

BENEFITS and TRADEOFFS

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

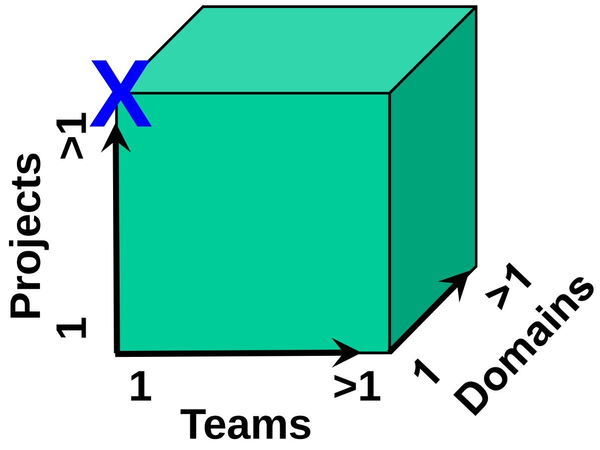
ANALYSIS

- Structure analysis: Inter-component connectivity analysis

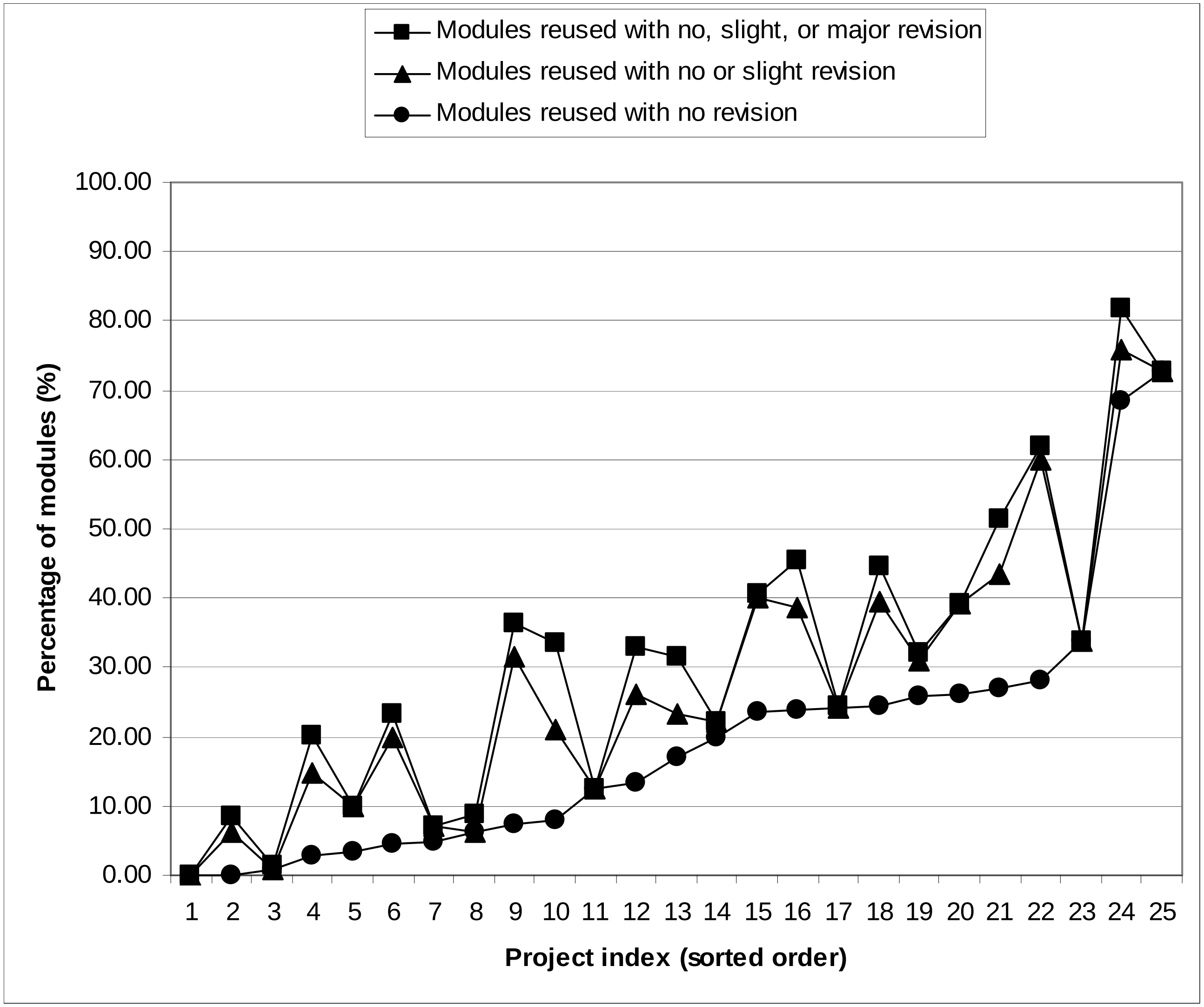
MODELING

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

SCALING DIMENSIONS



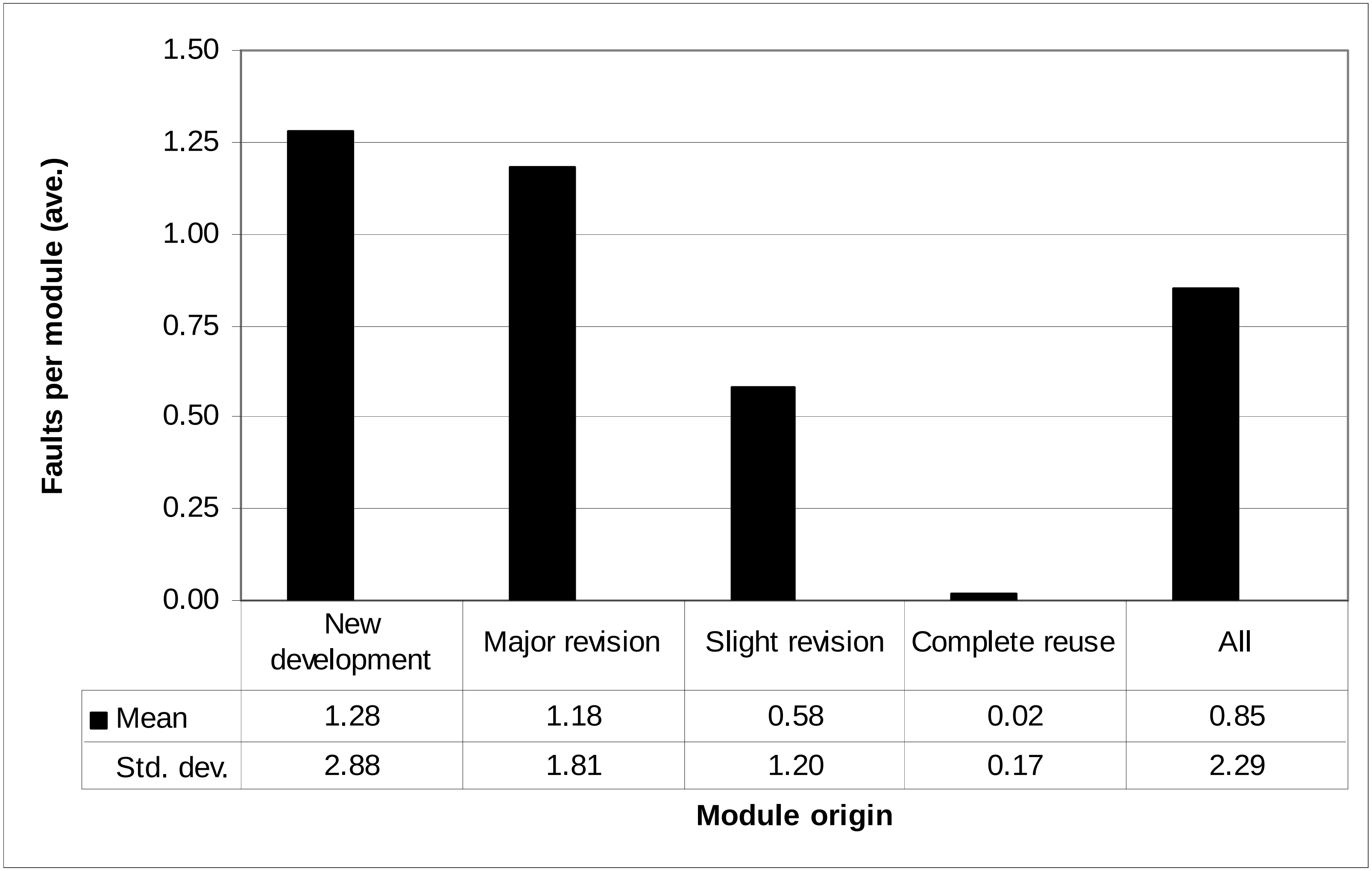
32% of Software Components are Either Reused or Modified from Previous Systems



- **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**

NORTHROP GRUMMAN

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



	New development	Major revision	Slight revision	Complete reuse	All
■ Mean	1.28	1.18	0.58	0.02	0.85
Std. dev.	2.88	1.81	1.20	0.17	2.29

- **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

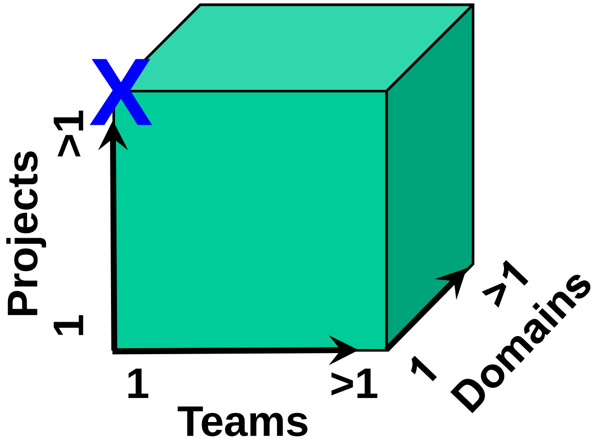
- Reuse analysis: Reconfigurable component-driven development
- **Structure analysis: Inter-component connectivity analysis**

Enables ?

MODELING

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

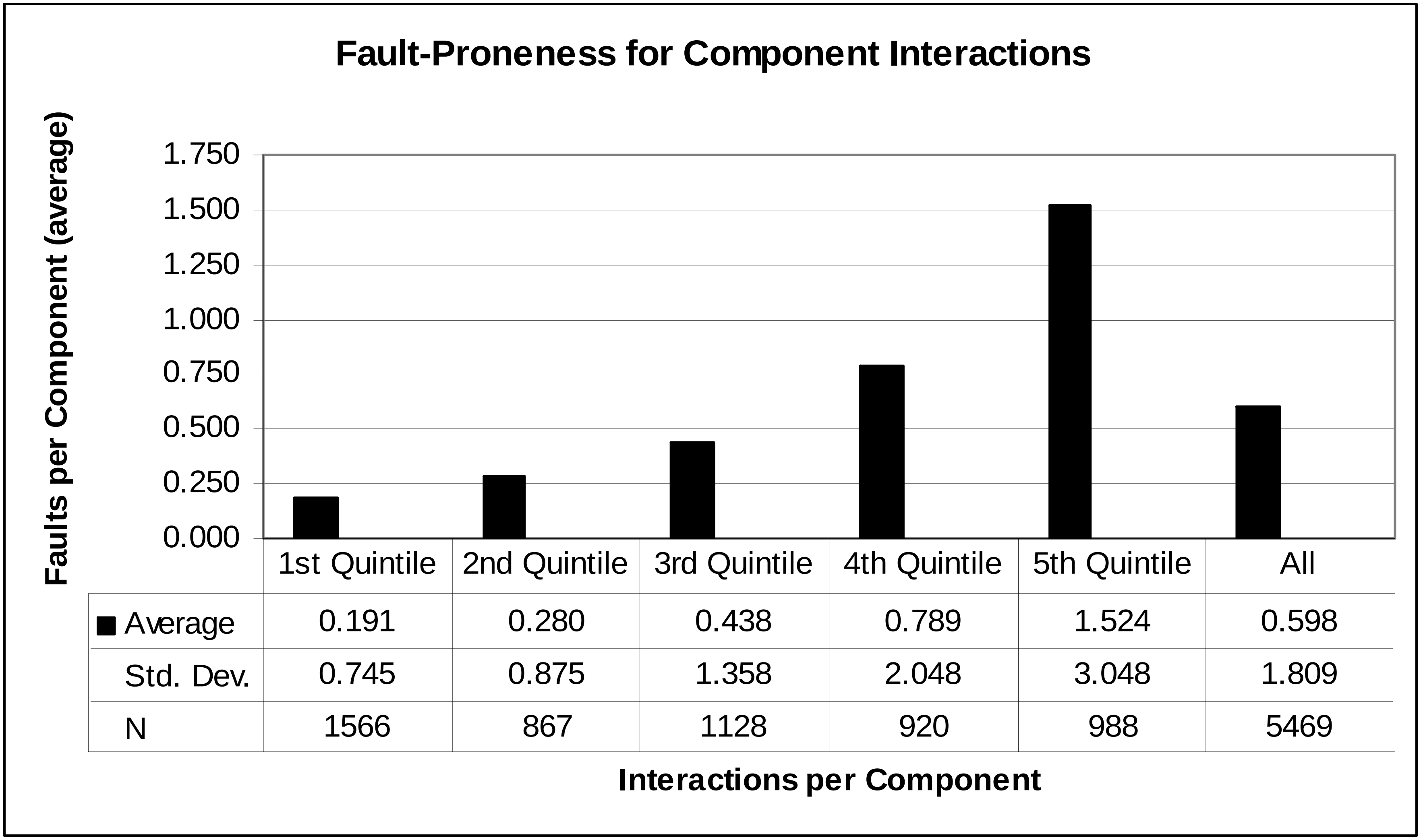
SCALING DIMENSIONS



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

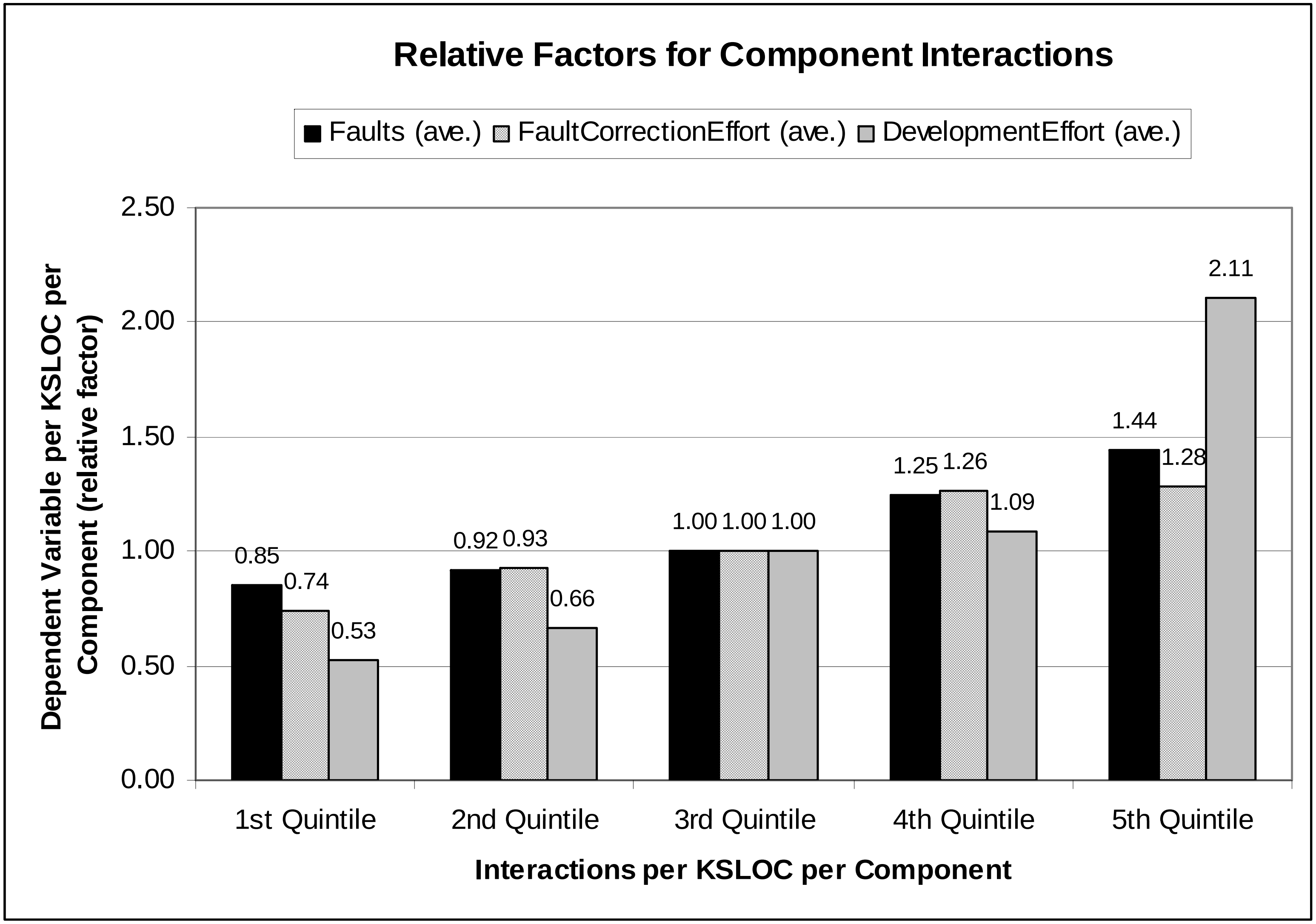


	1st Quintile	2nd Quintile	3rd Quintile	4th Quintile	5th Quintile	All
■ Average	0.191	0.280	0.438	0.789	1.524	0.598
Std. Dev.	0.745	0.875	1.358	2.048	3.048	1.809
N	1566	867	1128	920	988	5469

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

Absolute

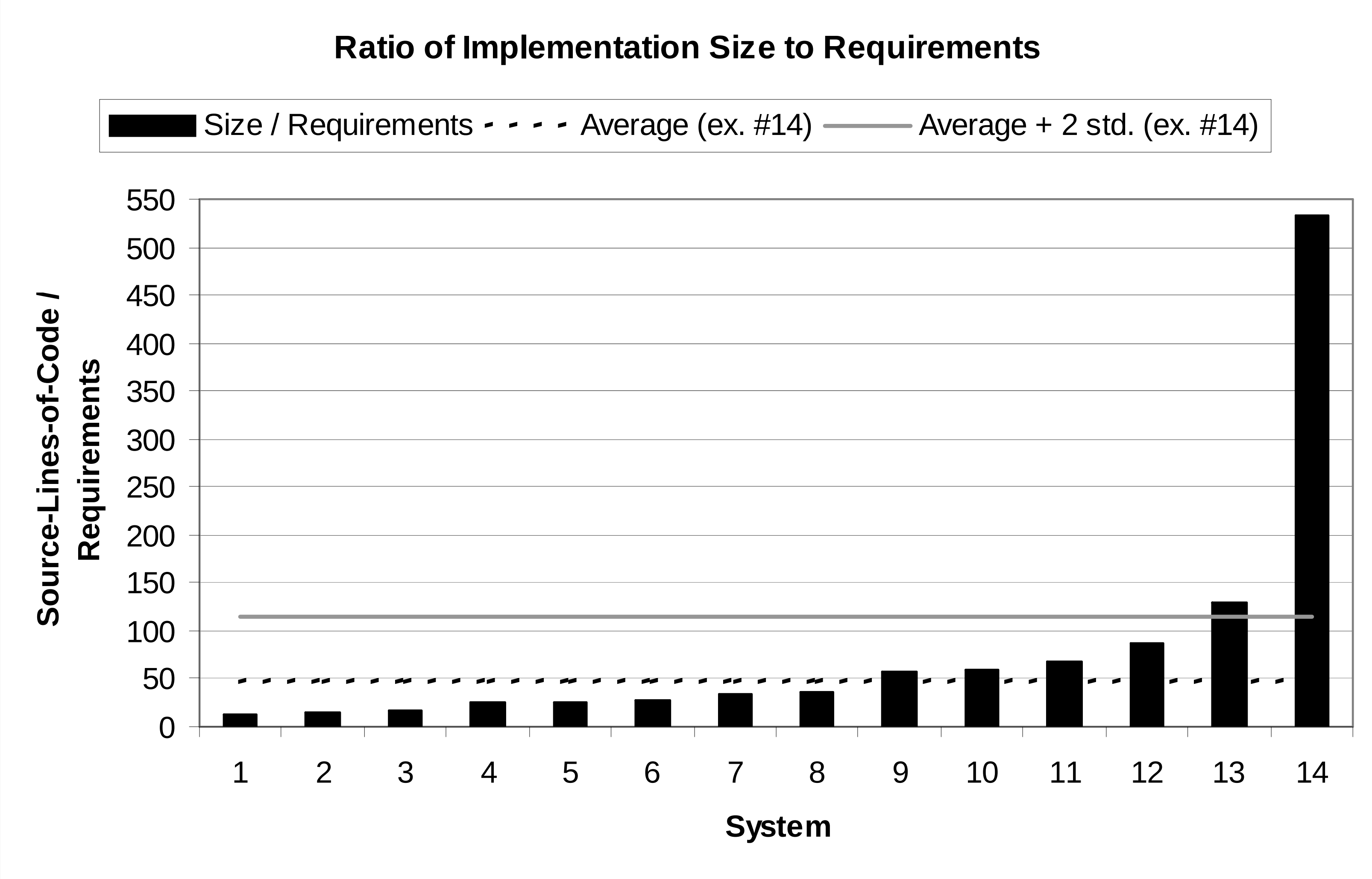
Analyses of Software Architectures Shows Fault Trends for Component Interaction Relative Factors



- **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

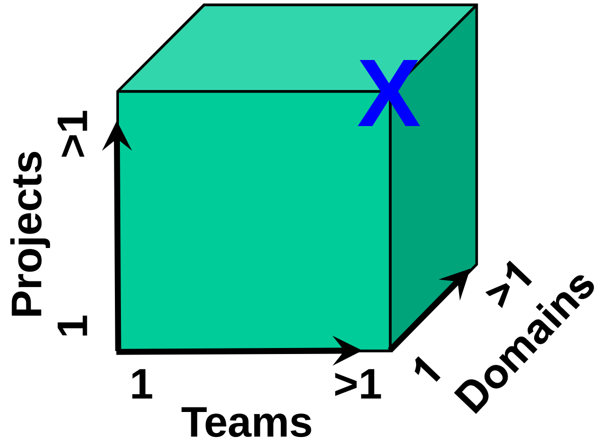
ANALYSIS

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

MODELING

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

SCALING DIMENSIONS



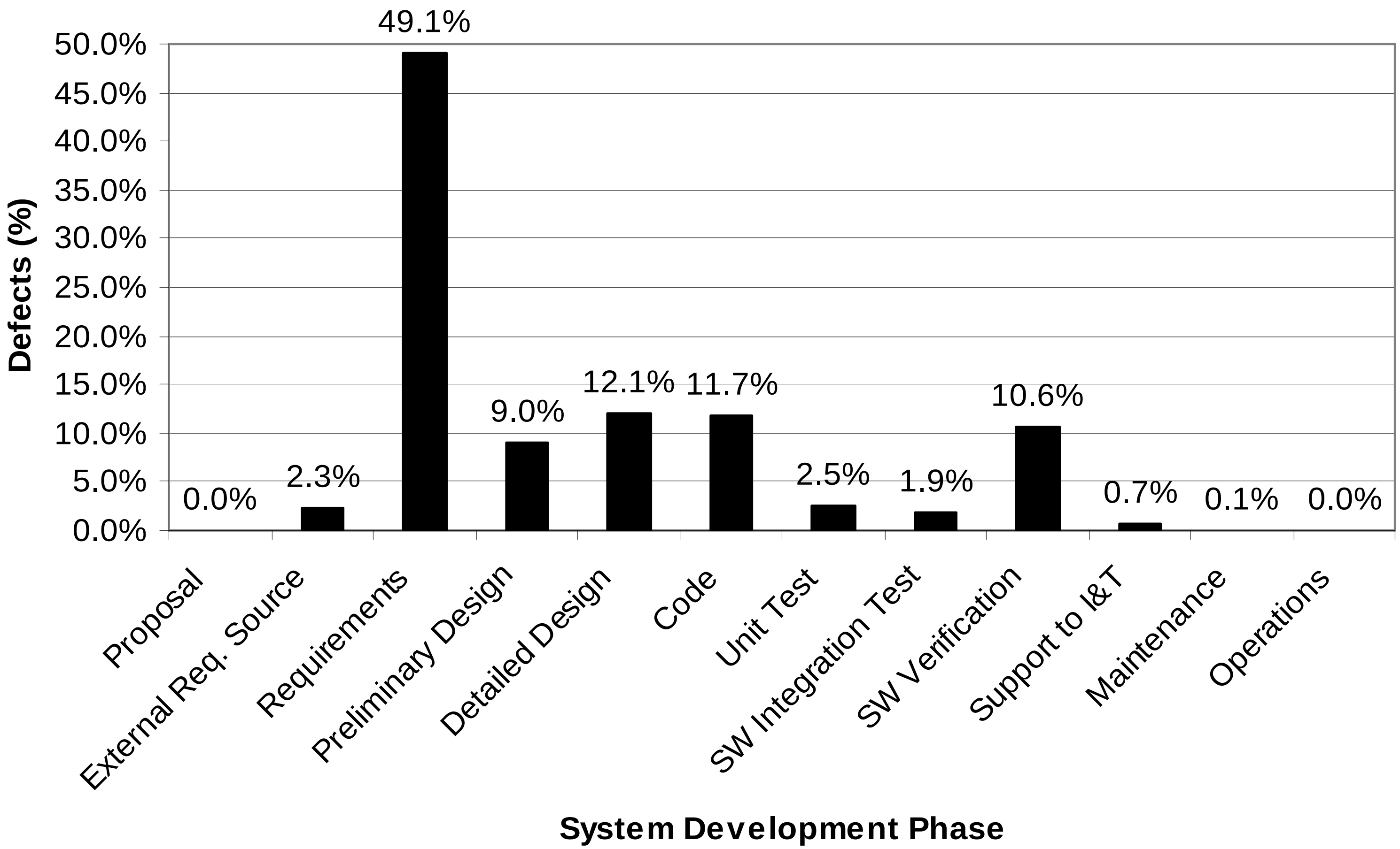
BENEFITS and TRADEOFFS

Enables?

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

NORTHROP GRUMMAN

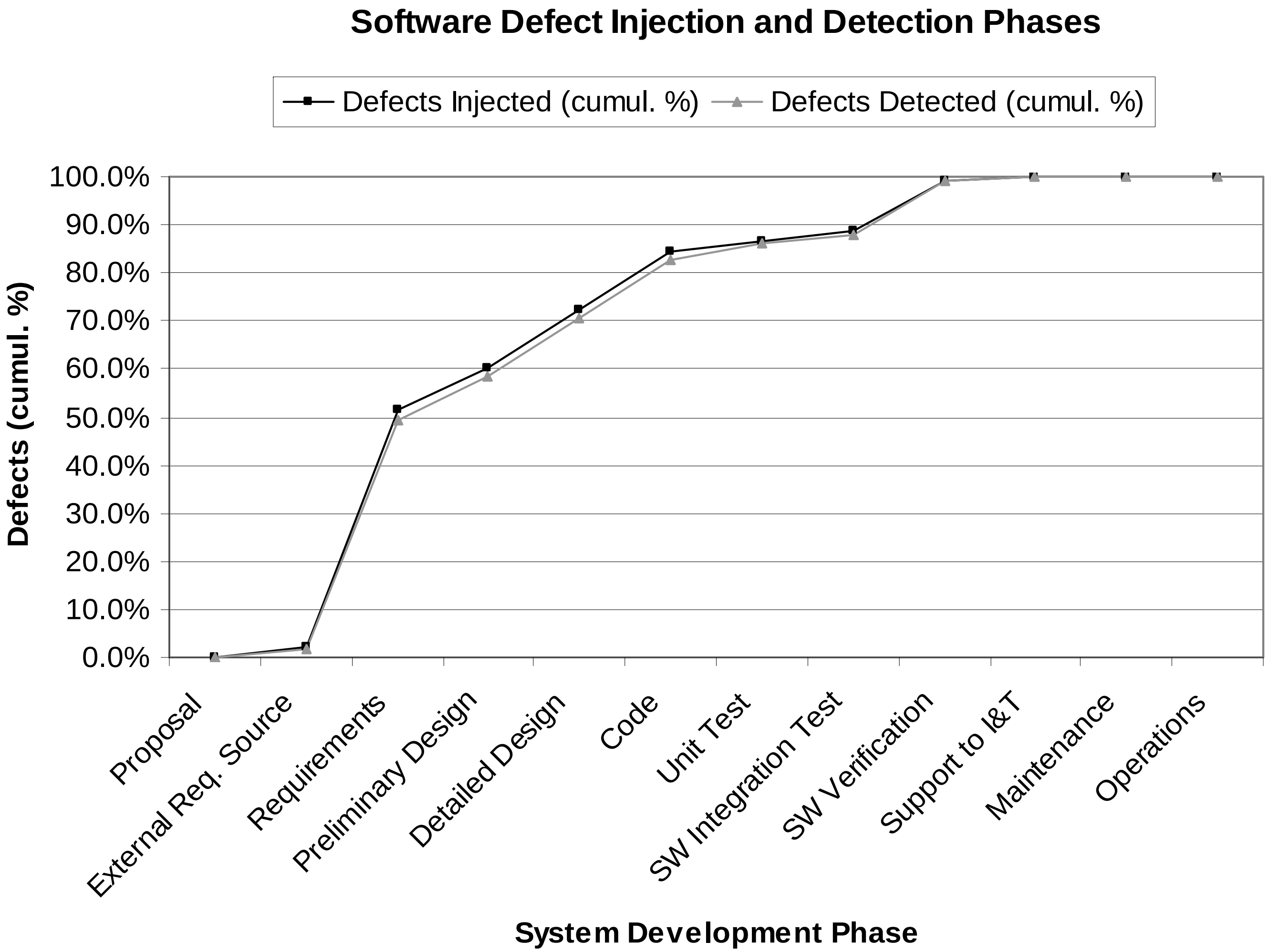
Analyses of Software Defect Injection Phases Reveals Distributions



Software Defect Injection Phase

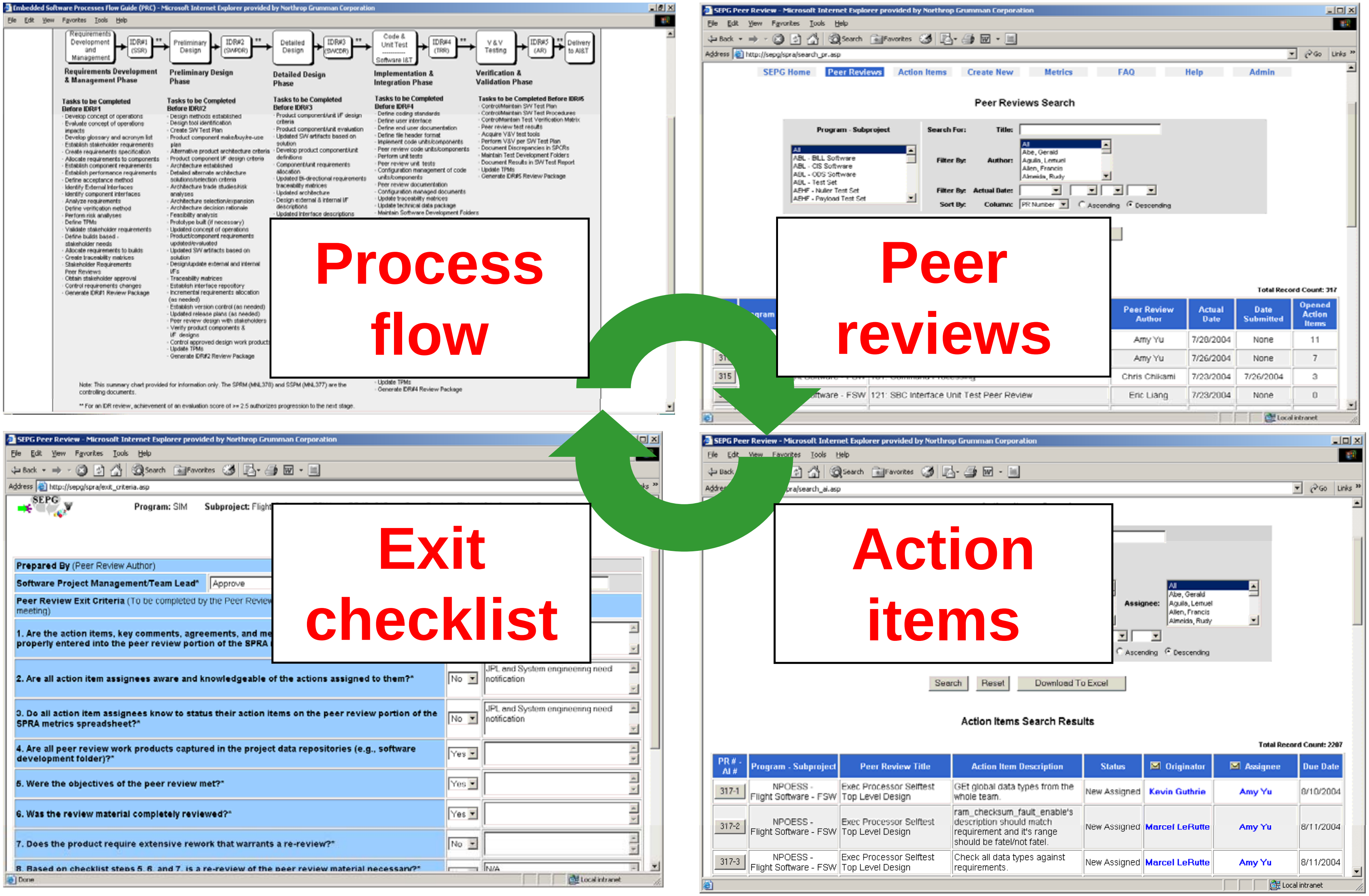
- Distribution of software defect injection phases based on using peer reviews across 12 system development phases
- 3418 defects, 731 peer reviews, 14 systems, 2.67 years
- 49% of defects injected during requirements phase

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



Total Record Count: 317

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

Program: SIM

Prepared By (Peer Review Author)

Software Project Management/Team Lead* Approve

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

2. Are all action item assignees aware and knowledgeable of the actions assigned to them?*	No	JPL and System engineering need notification
3. Do all action item assignees know to status their action items on the peer review portion of the SPRA metrics spreadsheet?*	No	JPL and System engineering need notification
4. Are all peer review work products captured in the project data repositories (e.g., software development folder)?*	Yes	
5. Were the objectives of the peer review met?*	Yes	
6. Was the review material completely reviewed?*	Yes	
7. Does the product require extensive rework that warrants a re-review?*	No	
8. Based on checklist steps 5, 6, and 7, is a re-review of the peer review material necessary?*		N/A

Action Items Search Results

Total Record Count: 2207

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 fatel/not fatel.	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

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

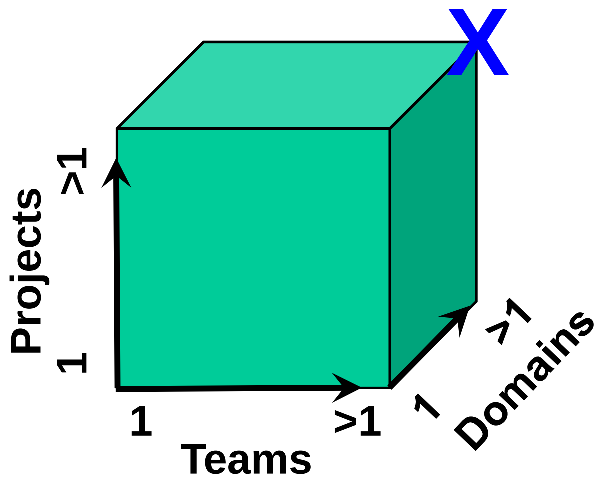
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis

MODELING

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

Enables?

SCALING DIMENSIONS

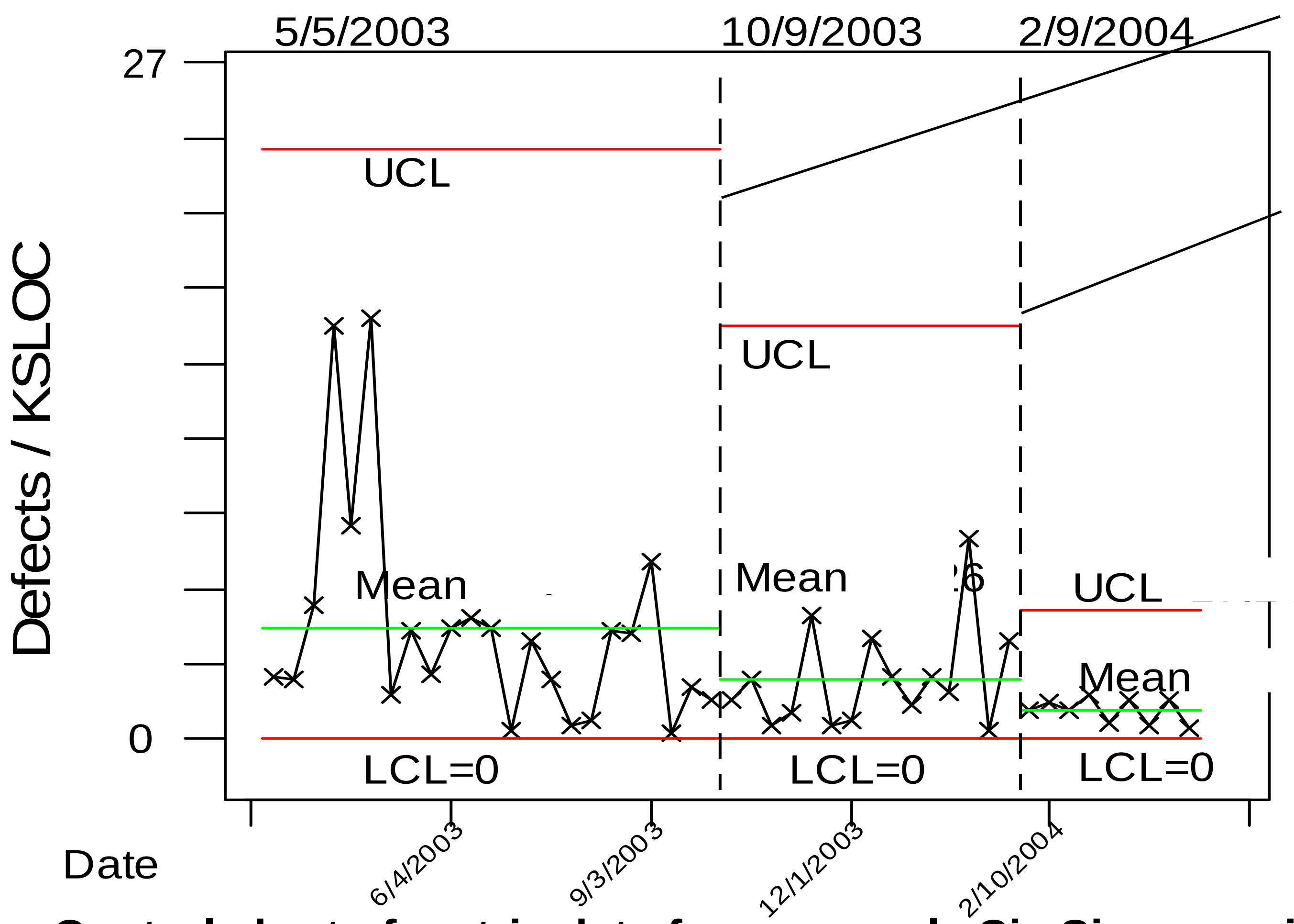


BENEFITS and TRADEOFFS

- 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



- **Six Sigma Project Introduced New Peer Review Process**
- **Provided Training on Process**

- **New Web-Based Peer Review Tool**
- **Provided Training on Tool**

These defects are action items resulting from peer reviews of software code and unit testing plans and results.

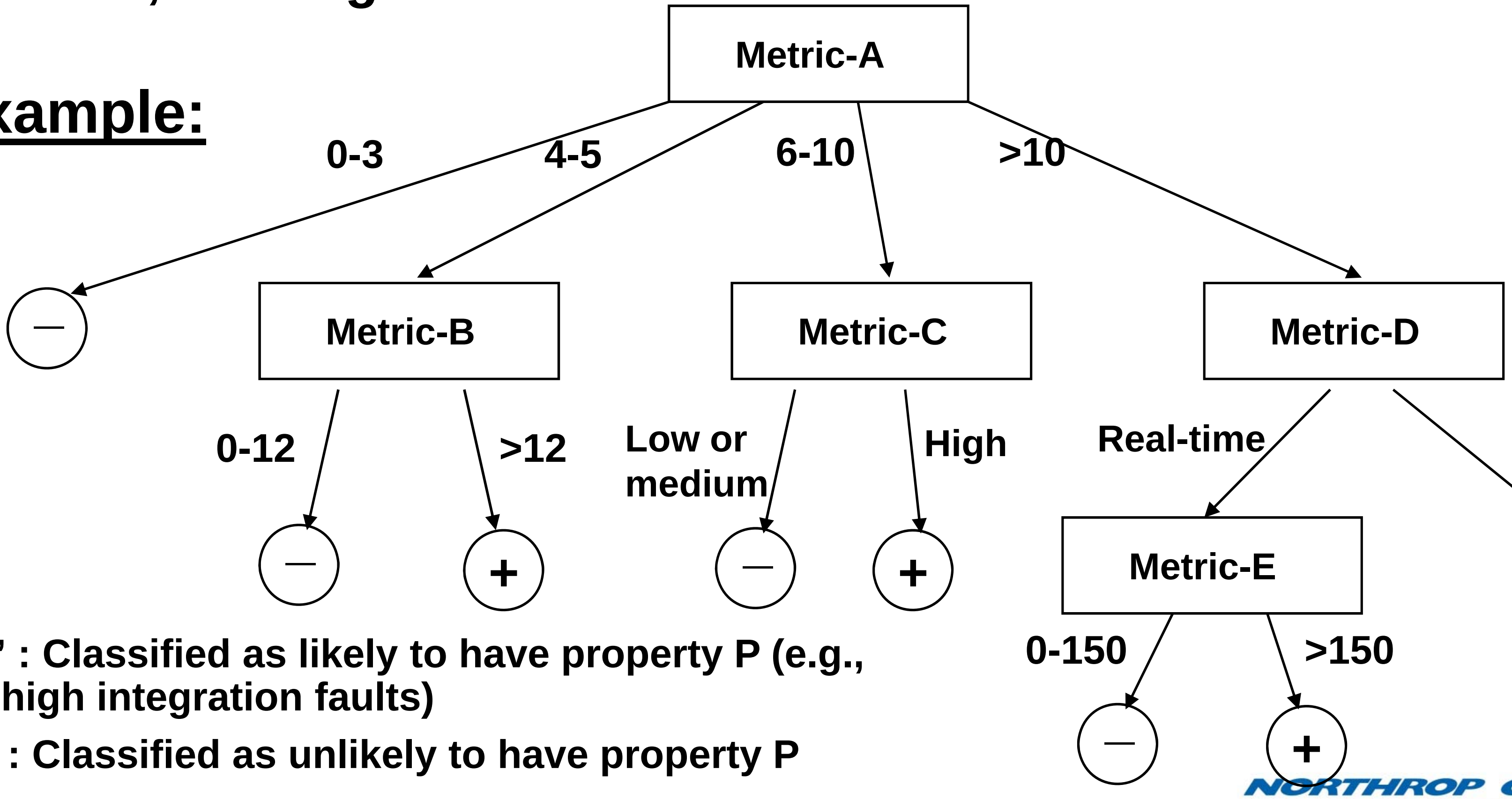
Data from 10 systems

- **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:



Metric-A

0-3

4-5

6-10

>10

—

Metric-B

Metric-C

Metric-D

0-12

>12

Low or medium

High

Real-time

Non-real time

—

+

—

+

Metric-E

—

0-150

>150

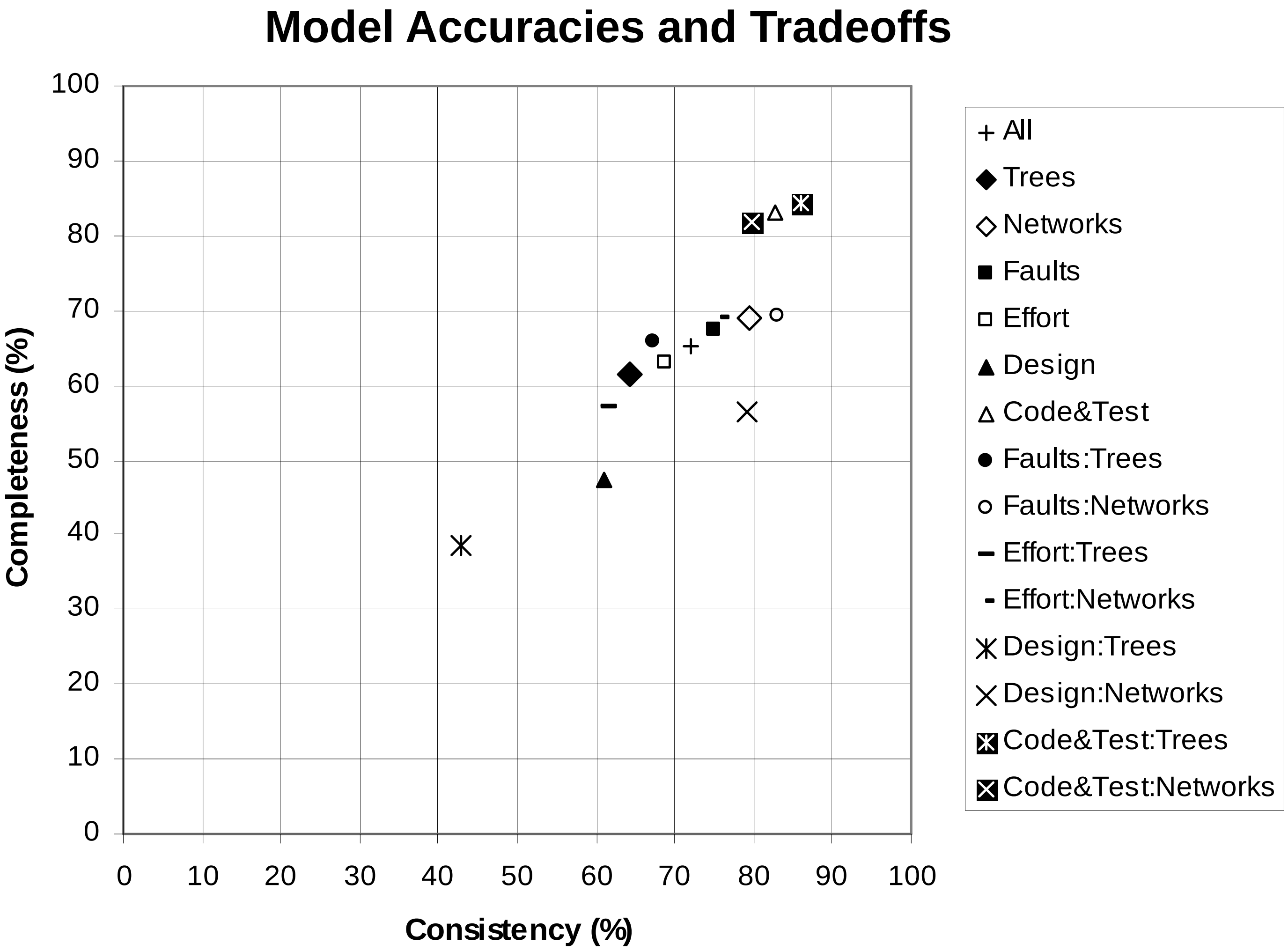
—

+

“+” : 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



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

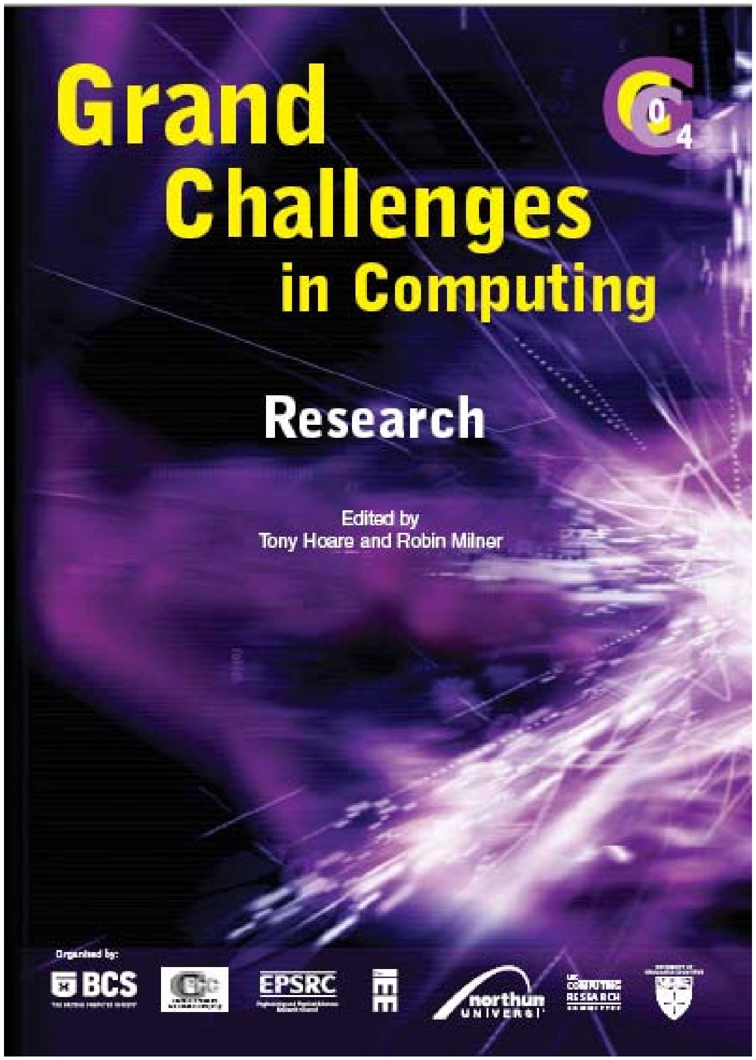
Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

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)



Grand Challenges in Computing

Research

Edited by
Tony Hoare and Robin Milner

Organised by:

BCS

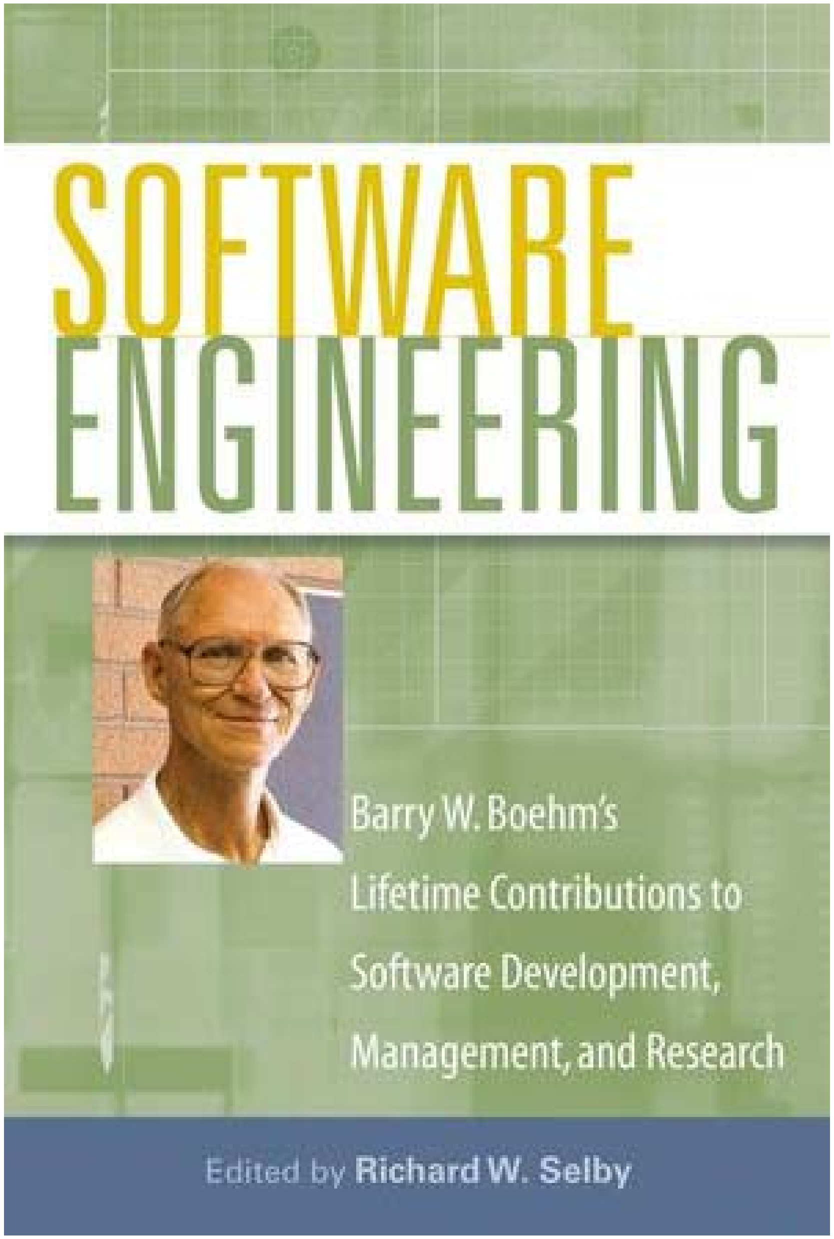
EPSRC

northumbria university

Contents

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

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



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

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