THE VALUE OF PERFORMANCE.
NORTHROP GRUMMAN

Development, Management, and Economics of Large-Scale Mission-Critical Systems

April 3, 2014

Rick Selby

Director of Engineering

Northrop Grumman Aerospace Systems

Redondo Beach, CA

Rick.Selby@NGC.com

Organizational Charter Focuses on Embedded Systems and Software Products

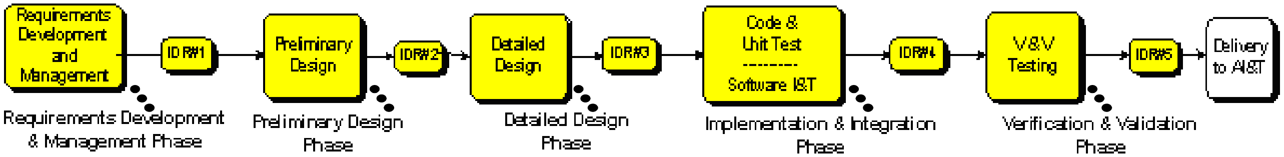
- Embedded software for
  - Advanced robotic spacecraft platforms
  - High-bandwidth satellite payloads
  - High-power laser systems
- Emphasis on both system management and payload software
- Reusable, reconfigurable software architectures and components
- Languages: O-O to C to asm
- CMMI Level 5 for 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

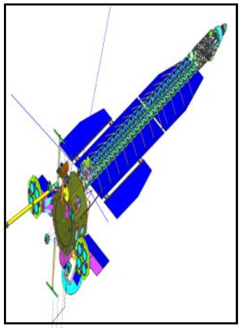
Requirements Development & Management Phase

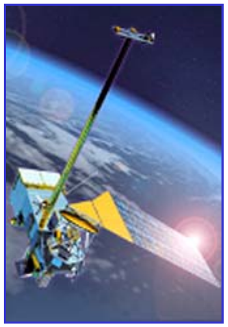
Preliminary Design Phase

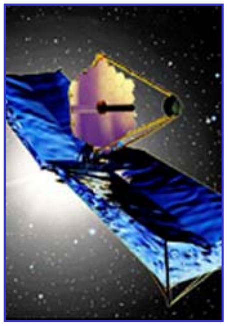
Detailed Design Phase

Implementation & Integration Phase

Verification & Validation Phase


Prometheus / JIMO


NPOESS


JWST


EOS Aqua/Aura


Chandra


GeoLITE

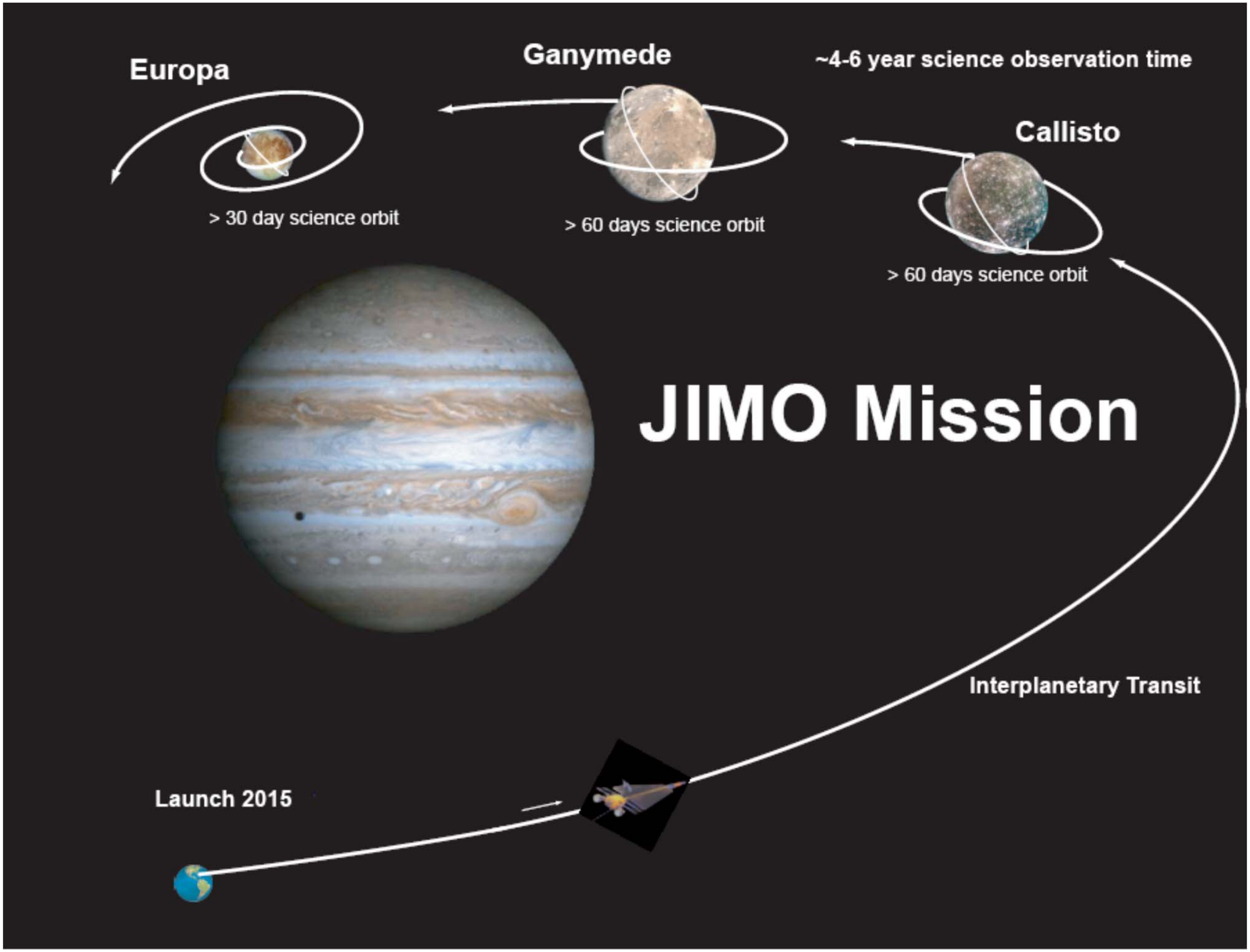

AEHF


MTHEL


Airborne Laser

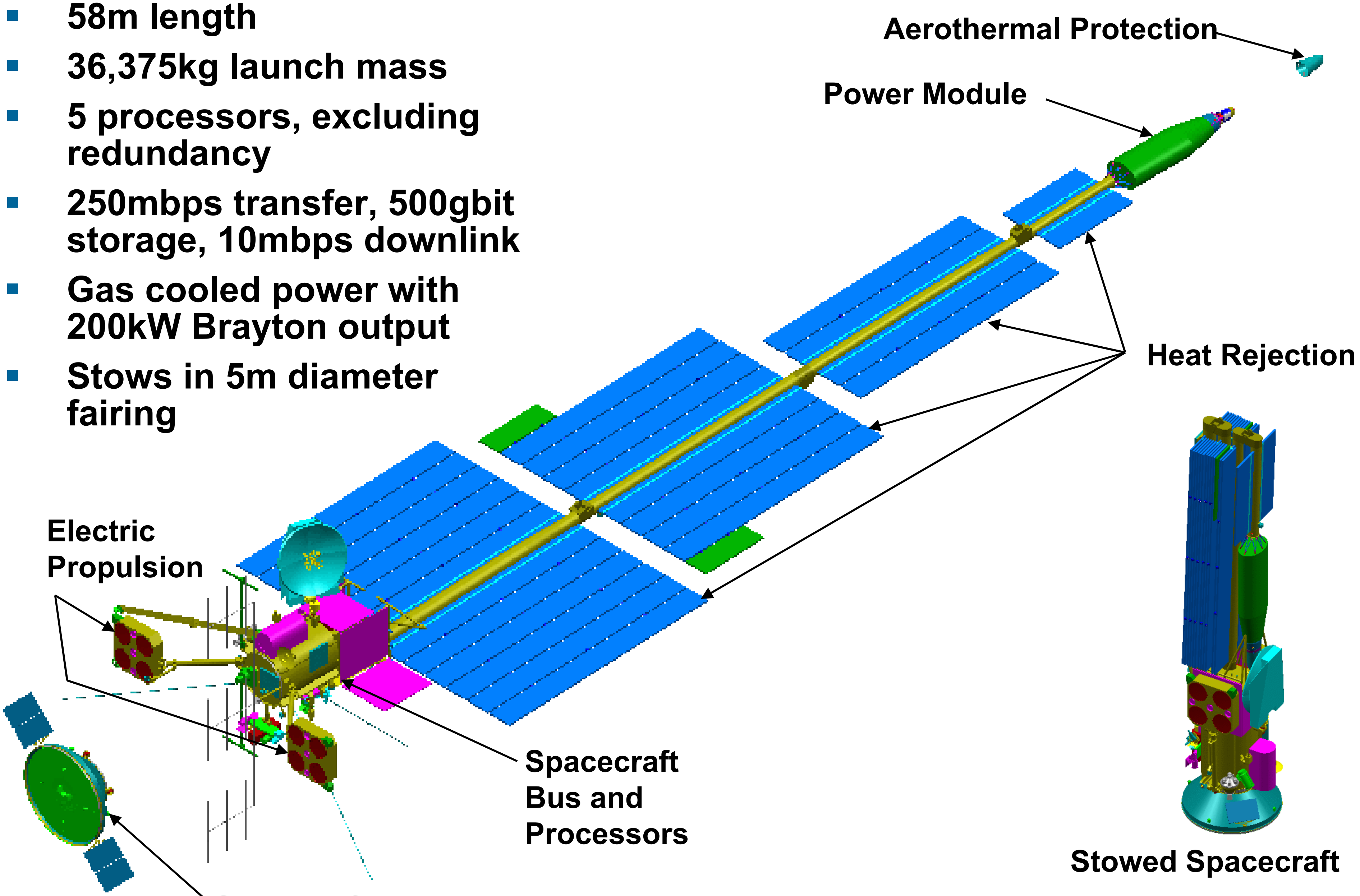

Restricted

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



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

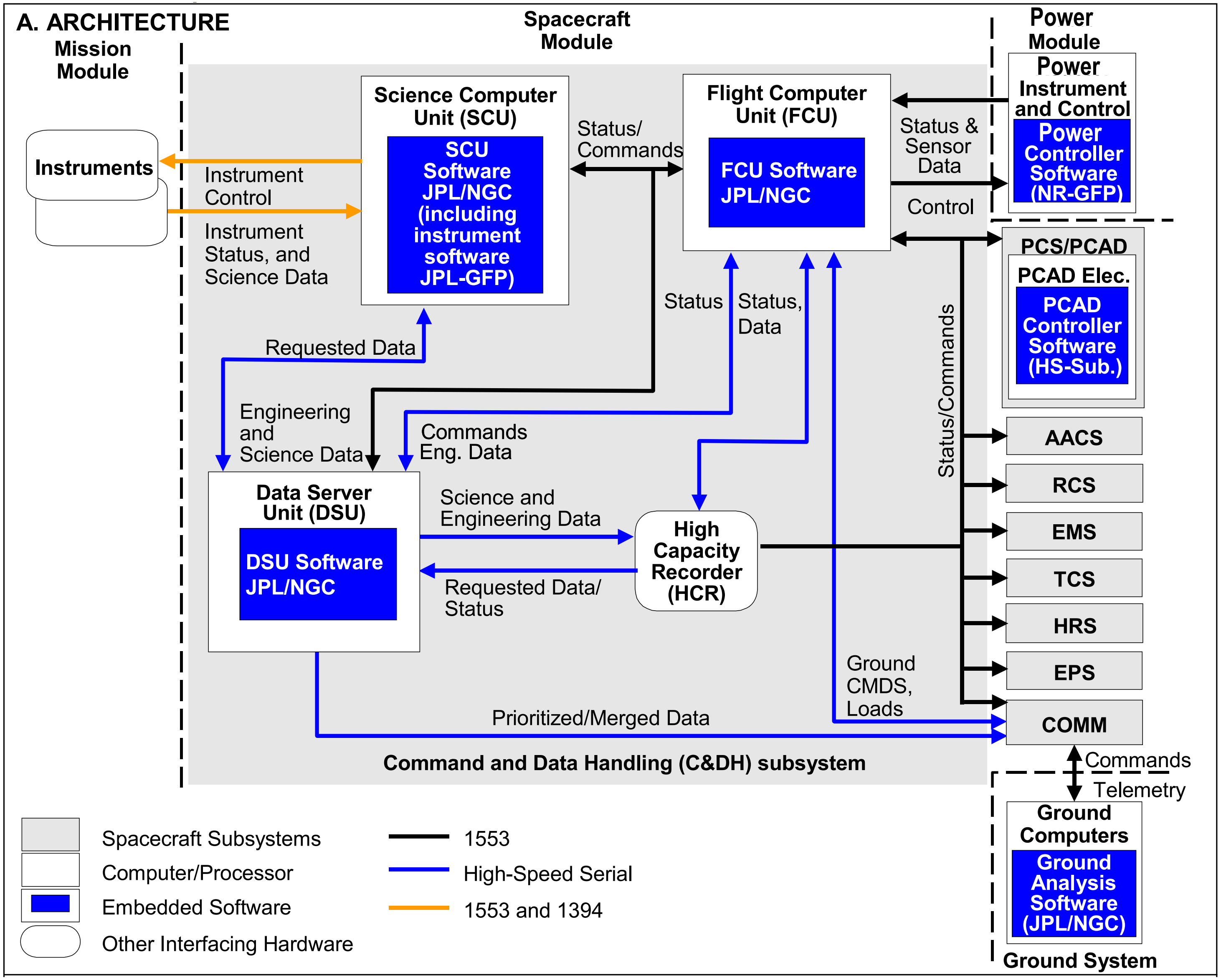
- Spacecraft configuration PB1
  - 58m length
  - 36,375kg launch mass
  - 5 processors, excluding redundancy
  - 250mbps transfer, 500gbit storage, 10mbps downlink
  - Gas cooled power with 200kW Brayton output
  - Stows in 5m diameter fairing



NORTHROP GRUMMAN

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



A. ARCHITECTURE

Mission Module

Spacecraft Module

Power Module

Instruments

Instrument Control

Instrument Status, and Science Data

Science Computer Unit (SCU)

SCU Software JPL/NGC (including instrument software JPL-GFP)

Status/ Commands

Flight Computer Unit (FCU)

FCU Software JPL/NGC

Status & Sensor Data

Control

Power Instrument and Control

Power Controller Software (NR-GFP)

PCS/PCAD

PCAD Elec.

PCAD Controller Software (HS-Sub.)

Requested Data

Status

Status, Data

Status/Commands

Engineering and Science Data

Commands Eng. Data

Data Server Unit (DSU)

DSU Software JPL/NGC

Science and Engineering Data

Requested Data/ Status

High Capacity Recorder (HCR)

AACS

RCS

EMS

TCS

HRS

EPS

COMM

Ground CMDS, Loads

Prioritized/Merged Data

Command and Data Handling (C&DH) subsystem

Commands

Telemetry

Ground Computers

Ground Analysis Software (JPL/NGC)

Ground System

▢	Spacecraft Subsystems	━	1553
▢	Computer/Processor	━	High-Speed Serial
■	Embedded Software	━	1553 and 1394
▢	Other Interfacing Hardware		

NORTHROP GRUMMAN

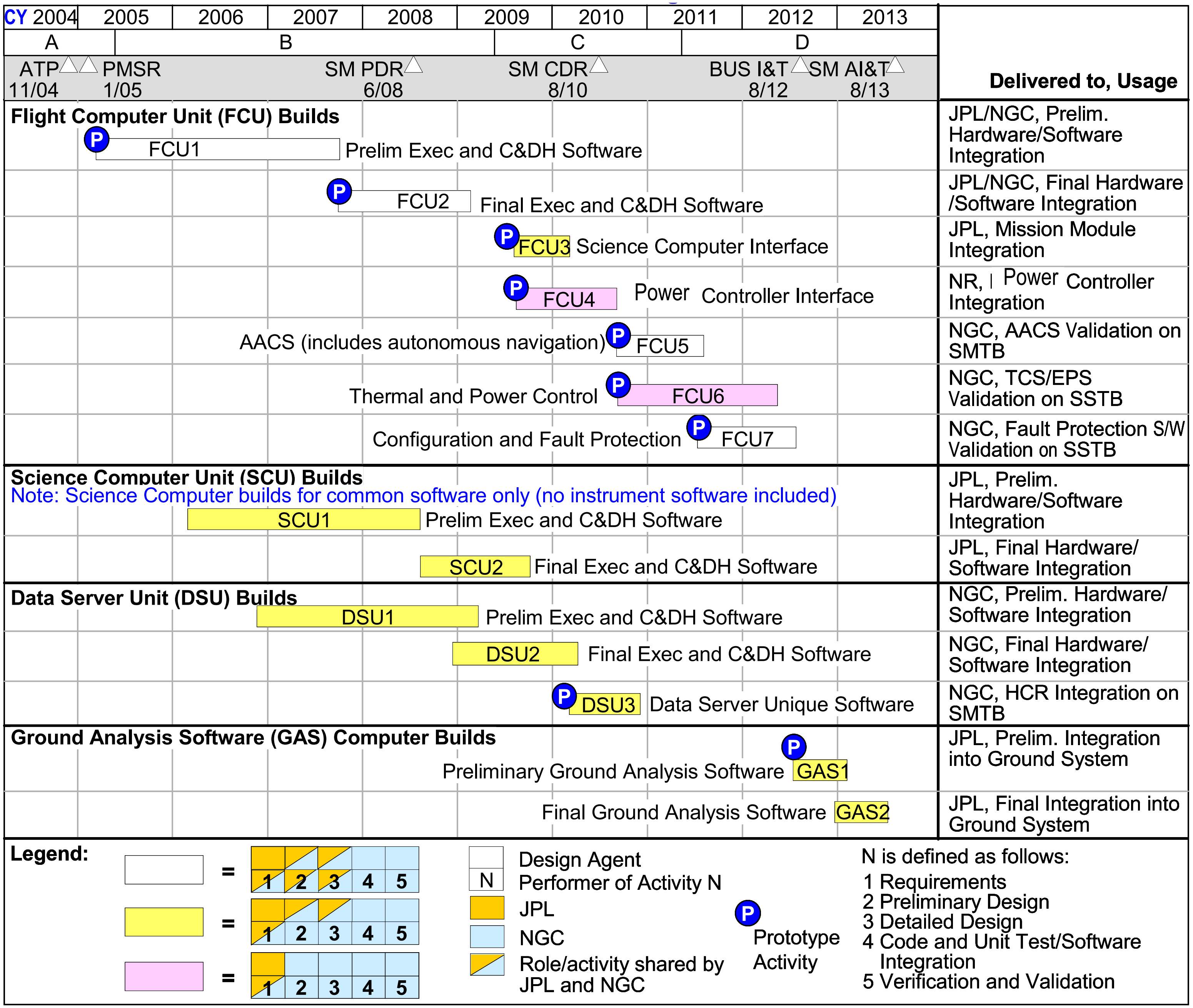
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

	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

Establish Embedded Systems and Software Design Principles: Strategies

- Adopt risk-driven incremental lifecycle with multiple software builds to improve visibility, accelerate delivery, and facilitate integration and test
- Share best-of-class software assets across organizations to leverage ideas and experience
- Define common software processes and tools to improve communication and reduce risk
- Conduct early tradeoff studies and end-to-end prototyping to validate key requirements, algorithms, and interfaces
- Design simple deterministic systems and analyze them with worst-case mindset to improve predictability
- Analyze system safety to minimize risk
- Conduct extensive reviews (intra-build peer reviews, five build-level review gates, higher level project reviews) and modeling, analysis, and execution (testbed conceptual, development, engineering, and flight models called CMs, DMs, EMs, and FMs) to enable thorough understanding, verification, and validation

Incremental Software Builds Deliver Early Capabilities and Accelerate Integration and Test



Build	Software	Delivered to, Usage
Flight Computer Unit (FCU) Builds		
FCU1 (P)	Prelim Exec and C&DH Software	JPL/NGC, Prelim. Hardware/Software Integration
FCU2 (P)	Final Exec and C&DH Software	JPL/NGC, Final Hardware /Software Integration
FCU3 (P)	Science Computer Interface	JPL, Mission Module Integration
FCU4 (P)	Power Controller Interface	NR, Power Controller Integration
FCU5 (P)	AACS (includes autonomous navigation)	NGC, AACS Validation on SMTB
FCU6 (P)	Thermal and Power Control	NGC, TCS/EPS Validation on SSTB
FCU7 (P)	Configuration and Fault Protection	NGC, Fault Protection S/W Validation on SSTB
Science Computer Unit (SCU) Builds		
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 (P)	Data Server Unique Software	NGC, HCR Integration on SMTB
Ground Analysis Software (GAS) Computer Builds		
GAS1 (P)	Preliminary Ground Analysis Software	JPL, Prelim. Integration into Ground System
GAS2	Final Ground Analysis Software	JPL, Final Integration into Ground System

Note: Science Computer builds for common software only (no instrument software included)

Legend:

☐ = 1 2 3 4 5

🟨 = 1 2 3 4 5

🟪 = 1 2 3 4 5

☐ Design Agent

N Performer of Activity N

🟧 JPL

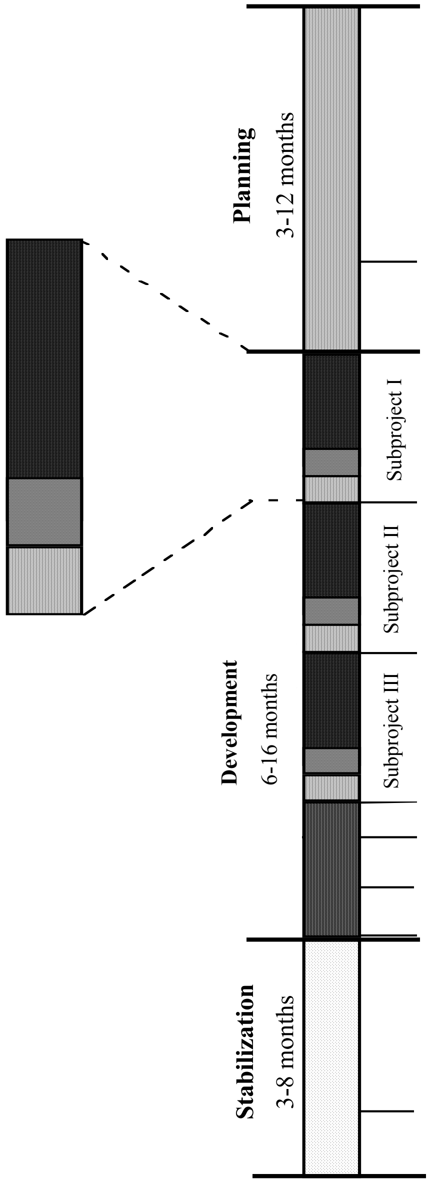
🟦 NGC

◩ 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

Synchronize-and-Stabilize Lifecycle Timeline and Milestones Enable Frequent Incremental Deliveries



Phases | **Timeline** | **Milestones** | **Major Reviews** | **Documents and Intermediate Activities**

Development Subproject
2-4 months
(1/3 of all features)

6-10 weeks
- Code and optimizations
- Testing and debugging
- Feature stabilization

2-5 weeks
- Integration
- Testing and debugging

2-5 weeks
- Buffer time

Milestone 0

Vision statement

Specification document

Specification review

Prototypes
Design feasibility studies
Testing strategy
Schedule

Schedule complete

Project review

Project plan approval

Implementation plan

Milestone I release

Milestone II release

Milestone III release

Visual freeze

Optimizations
Testing and debugging

Feature complete

Optimizations
Testing and debugging

Code complete

Internal testing
Buffer time
Beta testing
Buffer time

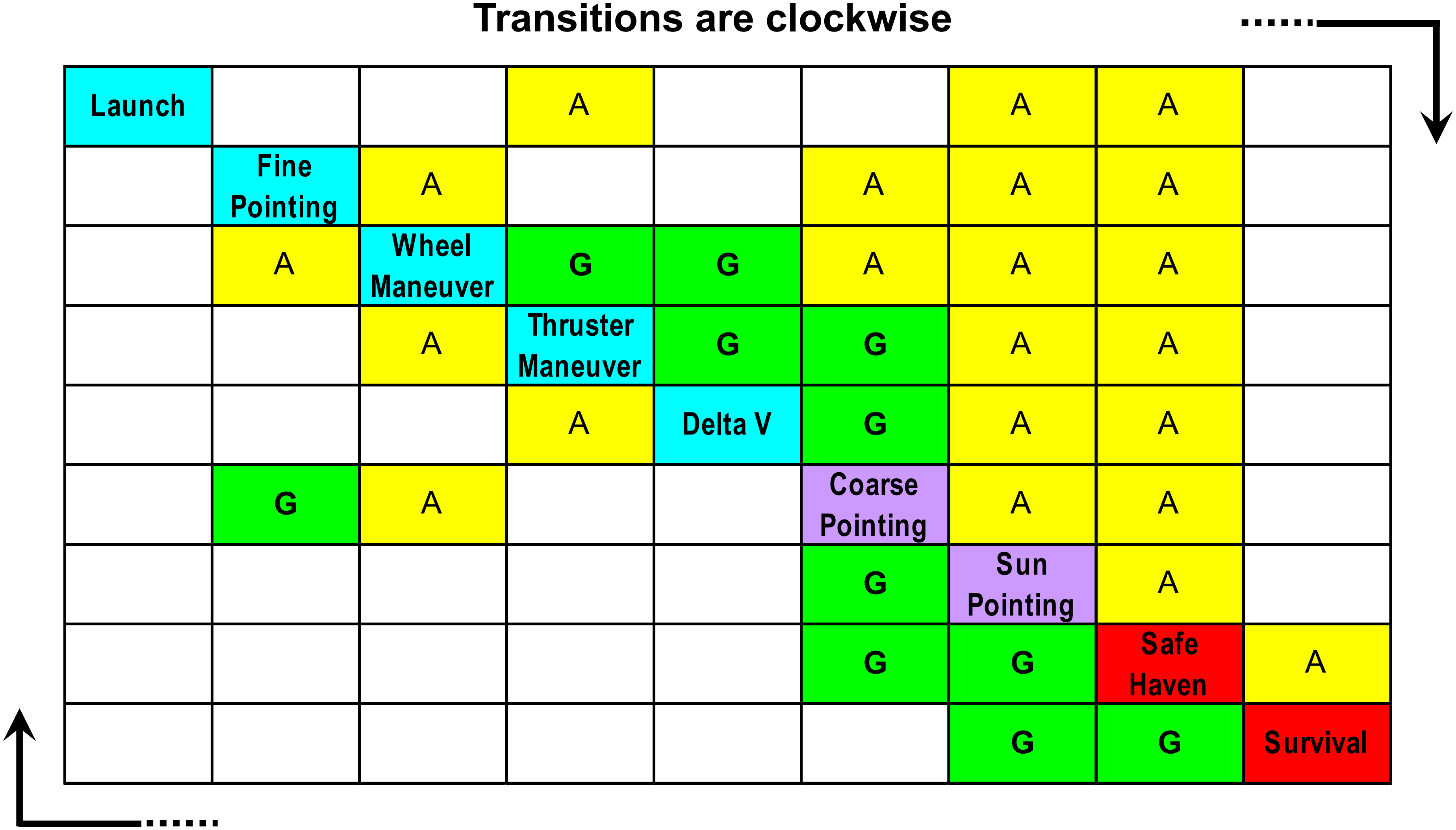
Zero bug release

Release to manufacturing
(Ship date)

Postmortem document

N^2 Tables Specify Valid Phase, State, or Mode Transitions to Facilitate Deterministic Designs

Transitions are clockwise



Launch			A			A	A	
	Fine Pointing	A			A	A	A	
	A	Wheel Maneuver	G	G	A	A	A	
		A	Thruster Maneuver	G	G	A	A	
			A	Delta V	G	A	A	
	G	A			Coarse Pointing	A	A	
					G	Sun Pointing	A	
					G	G	Safe Haven	A
						G	G	Survival

🟦 Nominal modes

🟪 Transition modes

🟥 Contingency modes

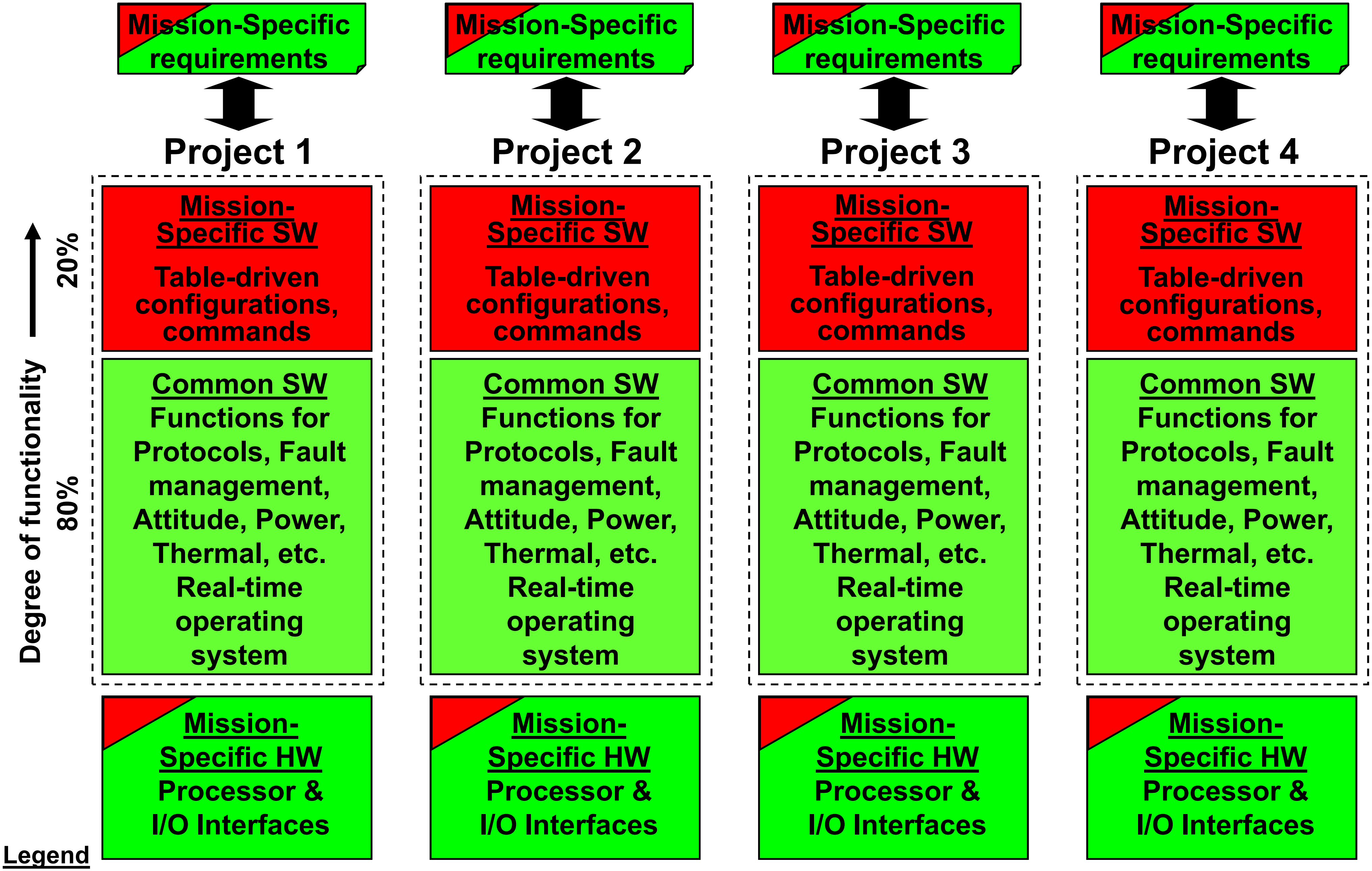
A Autonomous or Ground commanded

G Ground commanded only (real time only)

Establish Embedded Systems and Software Design Principles: Architectures

- <u>Partition functions across multi-processor</u> architecture (such as flight, science, data, power generation, power distribution) to distribute performance, allocate margins, and improve fault protection
- Define software "executive" <u>foundational layer that is common</u> across multi-processor architecture to enable reuse and flexibility
- Develop software using architectural simplicity, <u>table-driven designs</u>, deterministic behavior, and common interfaces to improve verifiability and predictability
- Adopt <u>high-level command sequencing "macro language"</u> for non-software personnel, such as system engineers, to use, typically structured as table-driven templates of commands and parameters, to improve specifiability and verifiability
- Define simple <u>deterministic synchronous control-loop designs</u> with well defined task structures (typically 3-4 levels), static resource allocation (such as no dynamic memory allocation), and predictable timing (such as minimizing interrupts) to improve understandability and verifiability
- Centralize system level <u>autonomy and fault protection</u> and distribute lower level autonomy and protection to appropriate control points to orchestrate system configurations, ensure timely isolations and responses, and support overall safety
- Dedicate <u>pathways for high-speed</u> data (such as from payload instruments to high capacity storage) to separate specialized processing and faults from core functionality (such as payload versus spacecraft)
- Adopt <u>large resource margins</u> (processor, memory, storage, bus) to accommodate contingencies and post-delivery changes

Common Requirements Enable Software Product Lines and Layered Architectures Across Projects



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

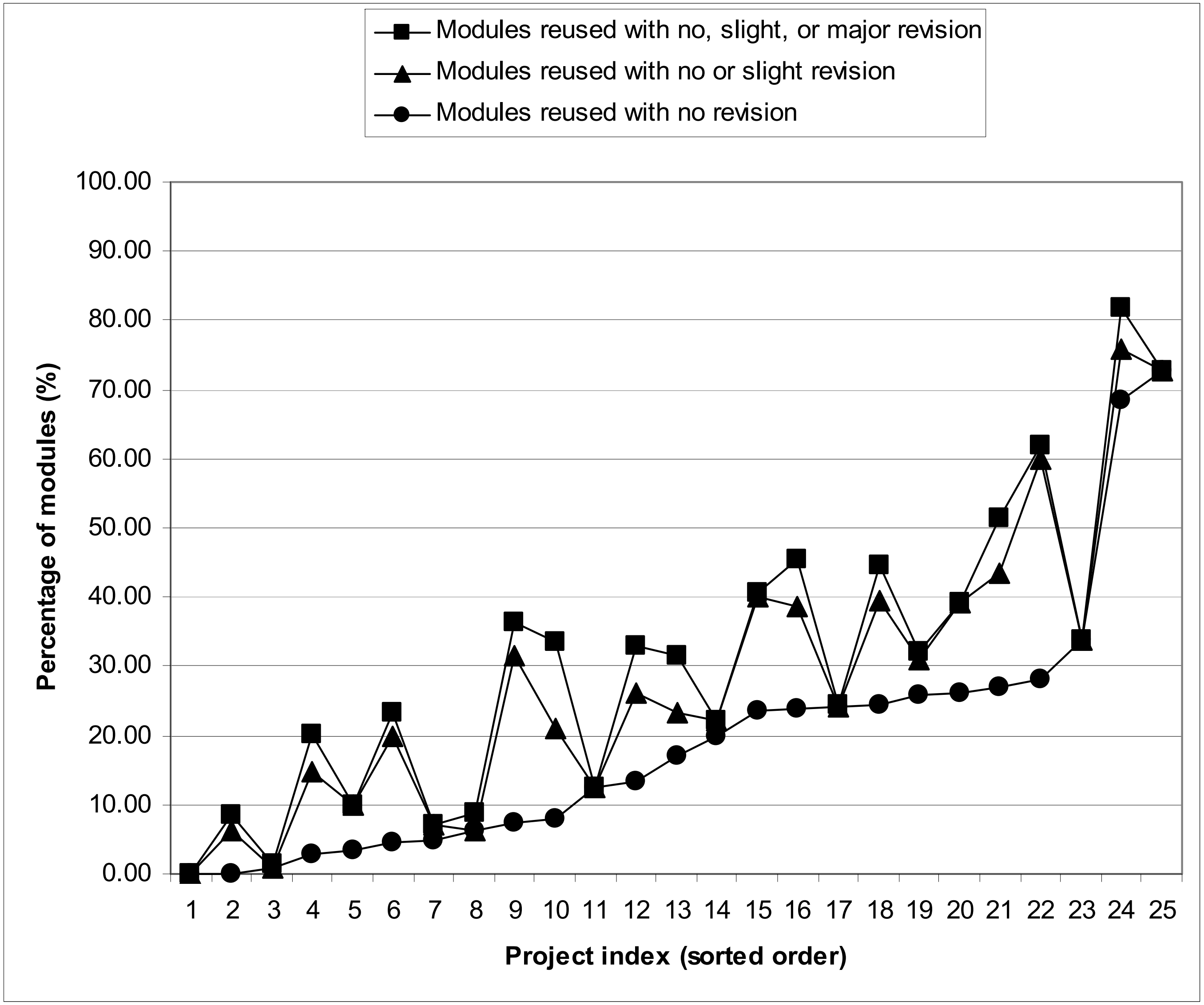
🟥 Mission-specific
🟩 Common across projects

Partition Software Functions Across Processors for Performance, Margins, and Fault Protection

Five-processor architecture provides partitioned functions, common executive layer, and growth margins

	Flight	**Science**	**Data**	**Power Generation**	**Power Distribution**
Processor-Specific	**SW Functions** Command sequencing Command execution Telemetry AACS Auto navigation Thermal control Power coordination Internal fault protection System fault protection	**SW Functions** Instrument control Instrument sequencing Instru. data processing Internal fault protection	**SW Functions** Recorder management Data storage control File/byte data protocol Data compression Internal fault protection	**SW Functions** Instrumentation Sensor control Drive control Coolant loop control Time-critical safing Internal fault protection	**SW Functions** Power conversion loop Sensor control Power distribution Array battery charging Health monitoring Internal fault protection
Common Executive	**SW Functions** Start-up ROM Initialization Processor self-test Device drivers Real-time O/S Time maintenance I/O management Memory load/dump Task management Shared data control Utilities & diagnostics	**SW Functions** Start-up ROM Initialization Processor self-test Device drivers Real-time O/S Time maintenance I/O management Memory load/dump Task management Shared data control Utilities & diagnostics	**SW Functions** Start-up ROM Initialization Processor self-test Device drivers Real-time O/S Time maintenance I/O management Memory load/dump Task management Shared data control Utilities & diagnostics	**SW Functions** Start-up ROM Initialization Processor self-test Device drivers Real-time O/S Time maintenance I/O management Memory load/dump Task management Shared data control Utilities & diagnostics	**SW Functions** Start-up ROM Initialization Processor self-test Device drivers Real-time O/S Time maintenance I/O management Memory load/dump Task management Shared data control Utilities & diagnostics
Margins	**>50%**	**>50%**	**>50%**	**>50%**	**>50%**

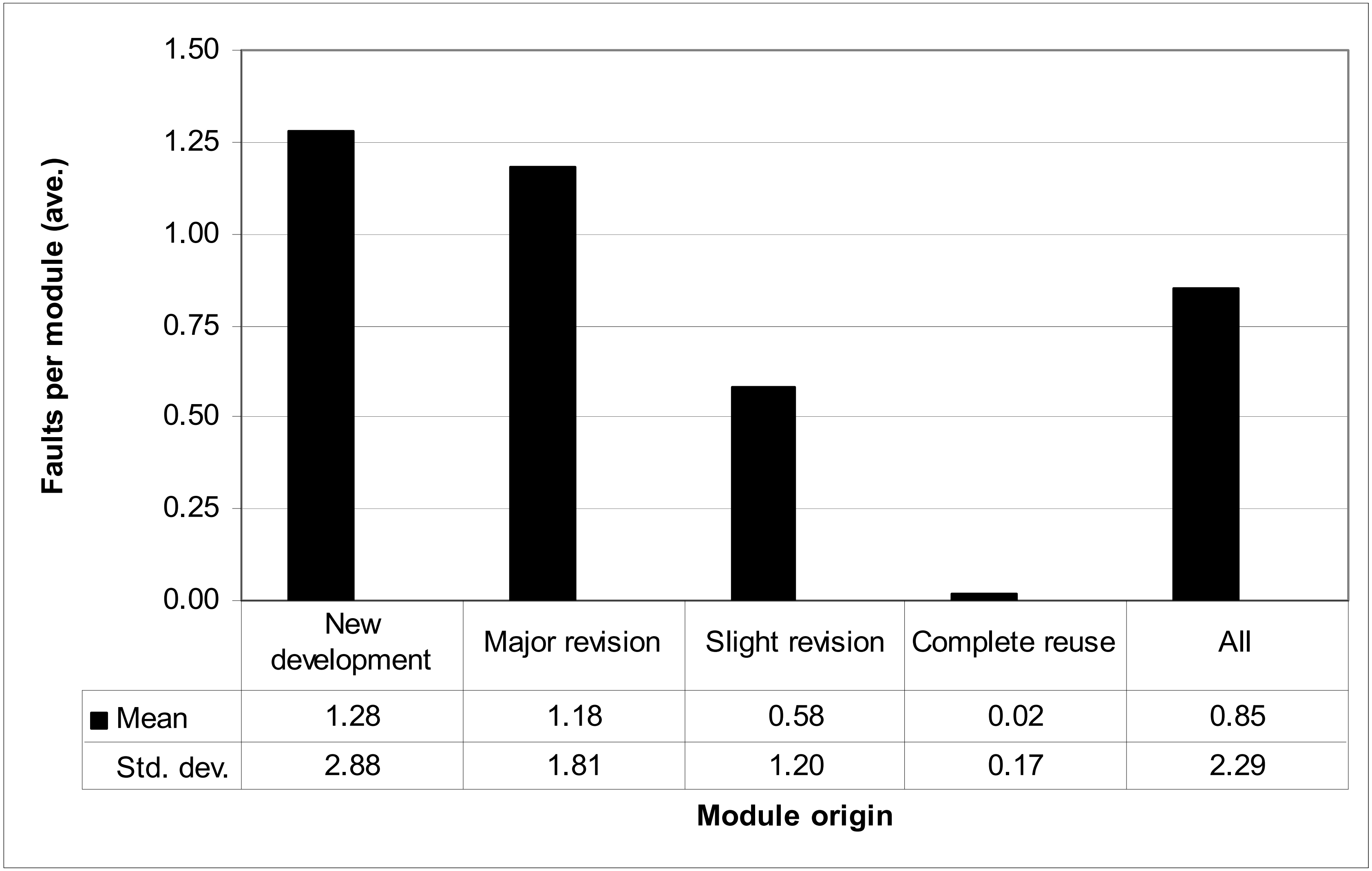
32% of Software Components are Either Reused or Modified from Previous Systems



- **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

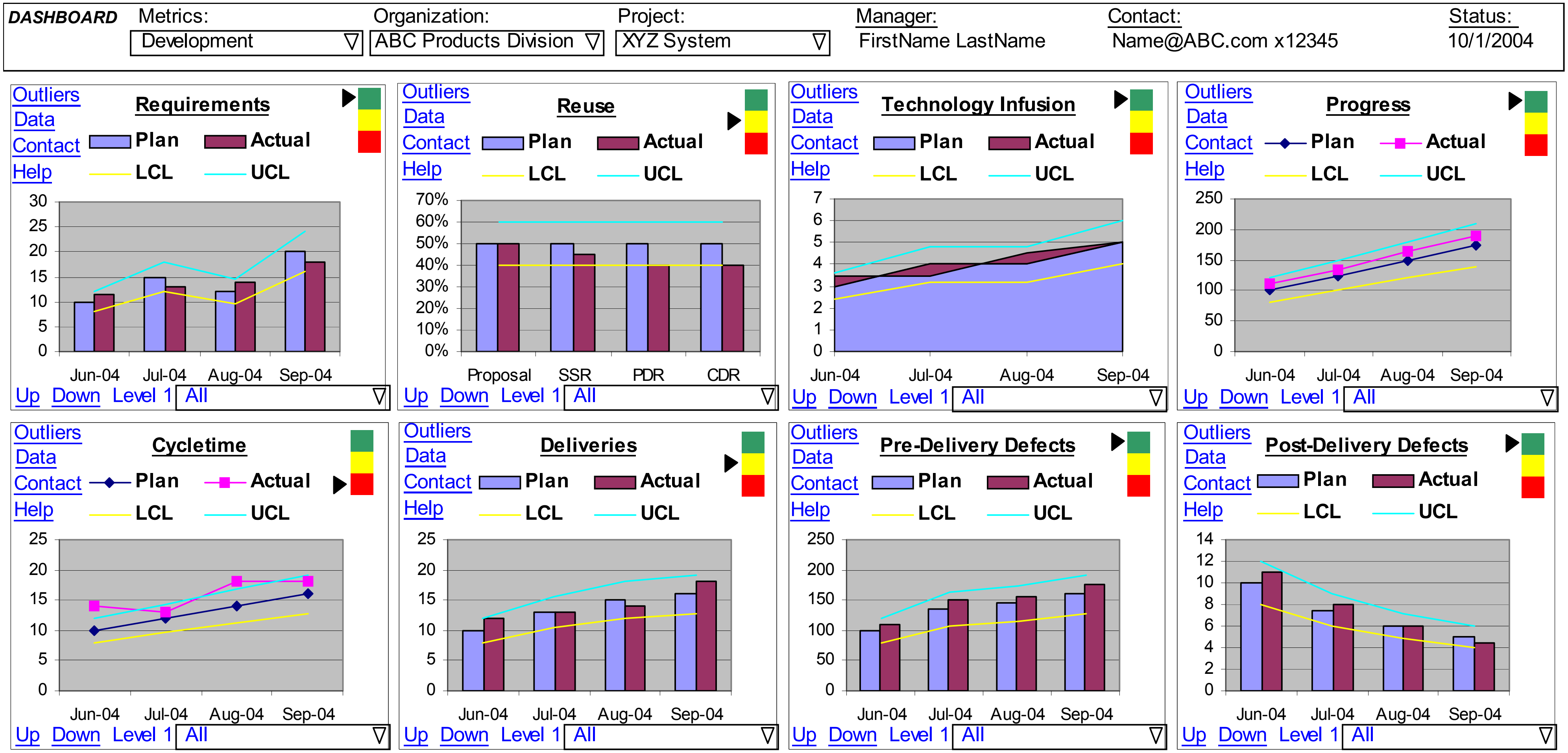
- **Data from 25 NASA systems**
- **Overall difference is statistically significant ($\alpha < .0001$). Number of components (or modules) in each category is: 1629, 205, 300, 820, and 2954, respectively.**

Establish Embedded Systems and Software Design Principles: Techniques

- Flowdown requirements systematically from project, system (space, ground, launch, etc.), module (spacecraft, mission, etc.), segment (bus, software, etc.), subsystem/build, assembly, etc. to clarify functionality and accountability
- Identify a manageable number of "key driving requirements", where key is top-down mission-success and driving is bottom-up design-limiting, to prioritize attention and analysis
- Define user-perspective "mission threads" to focus modeling, end-to-end prototyping, and validation
- Formulate leading indicators to identify high-fault and high-effort system structure and components
- Specify "command" abstractions that define standalone command primitives with pre-conditions, atomic processing, resource constraints (such as timing), and post-conditions (such as data modified) to enable analysis and predictability
- Define and enforce "control points", such as centralized sequential command queue and explicit data dependency graphs for read/write of data shared across commands and sequences, to facilitate analysis and isolate faults
- Include built-in self-tests, invariants, and redundant calculations in implementations to help ensure accurate processing and isolate faults
- Compare executions of system models and software implementations automatically using toolsets to improve verification
- Apply workflow tools, checklists, statistical analyses, root cause analyses, and metric dashboards to improve repeatability, visibility, and preventability

NORTHROP GRUMMAN

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

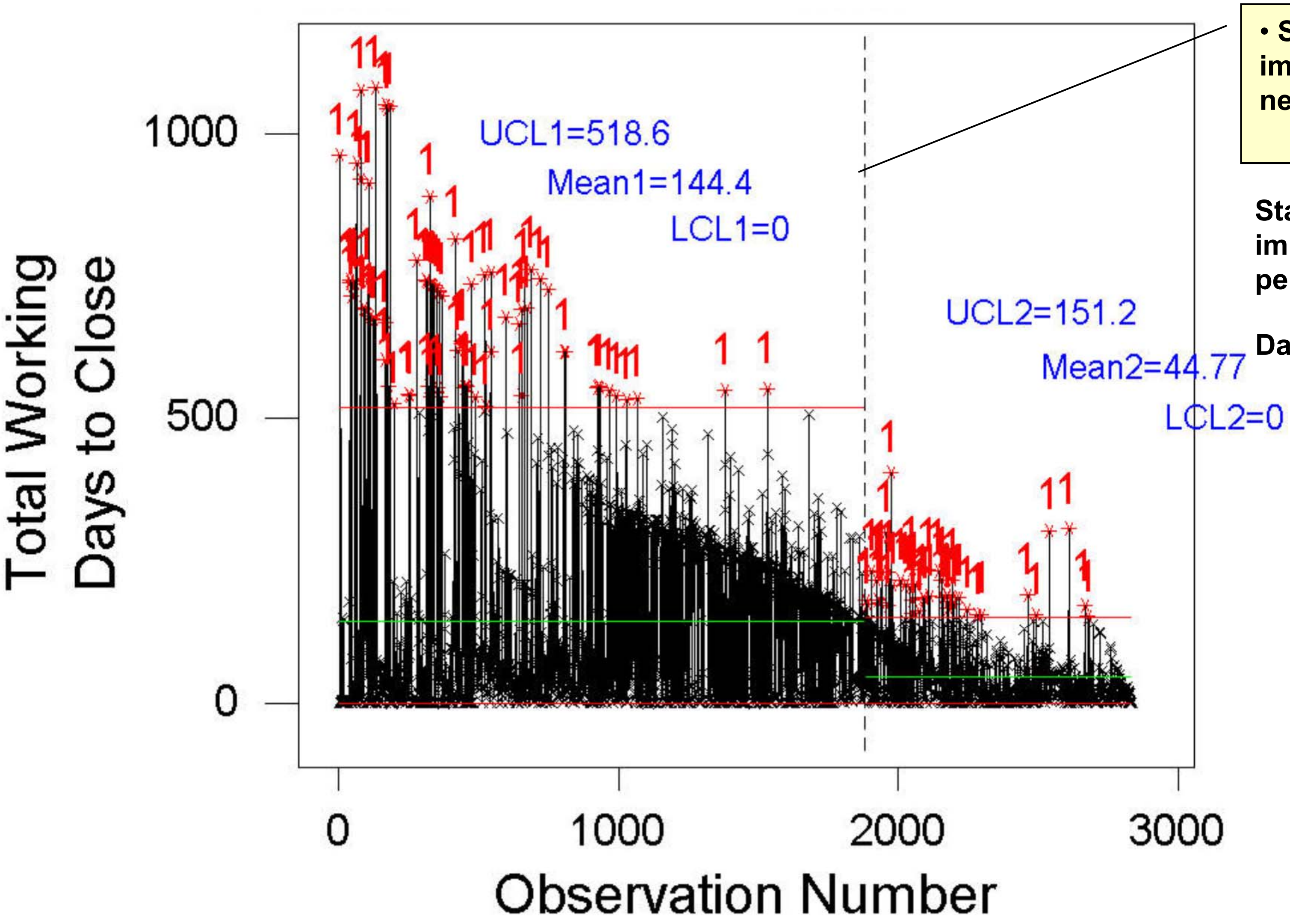


DASHBOARD

Metrics:	Organization:	Project:	Manager:	Contact:	Status:
Development ▽	ABC Products Division ▽	XYZ System ▽	FirstName LastName	Name@ABC.com x12345	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**

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



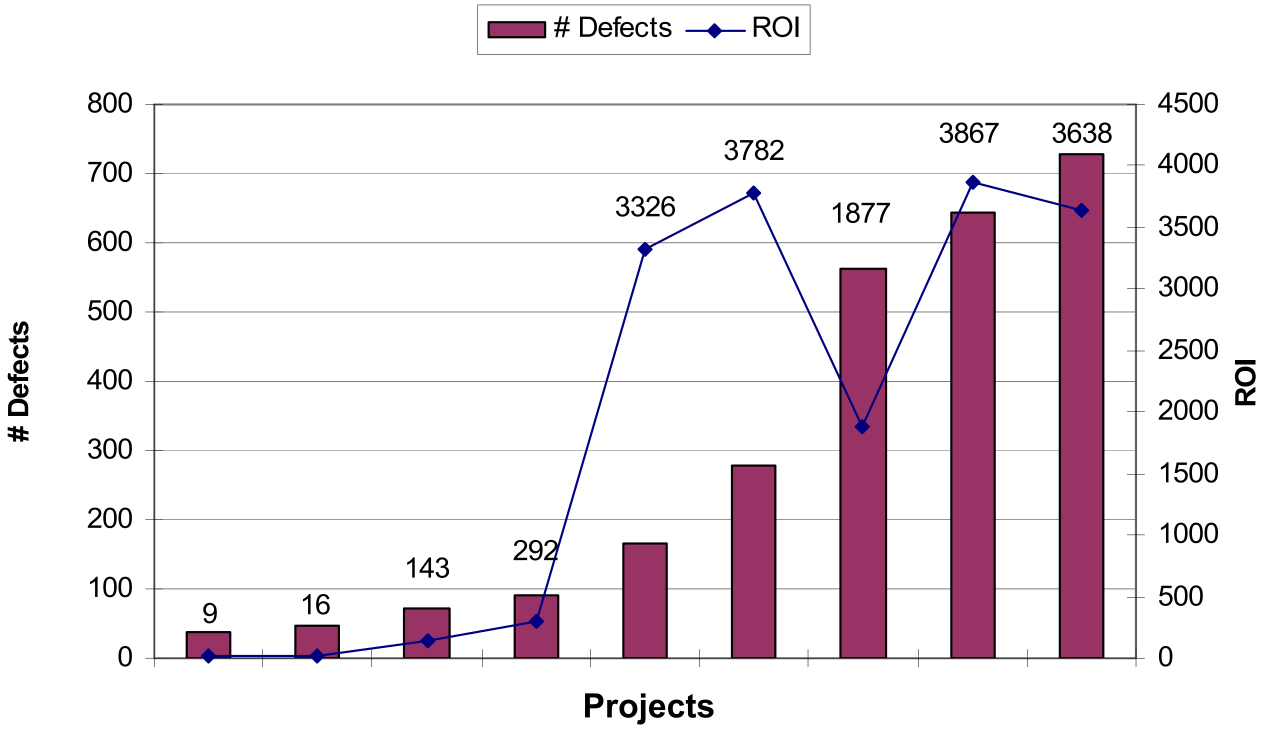
- Series of process improvements instituted and new control limits calculated

Statistically significant improvement in process performance

Data from one system

- **Control chart of metric data from example Six Sigma projects focusing on change request closure cycletime for software components**
- **Process improvements decreased variances and decreased means**

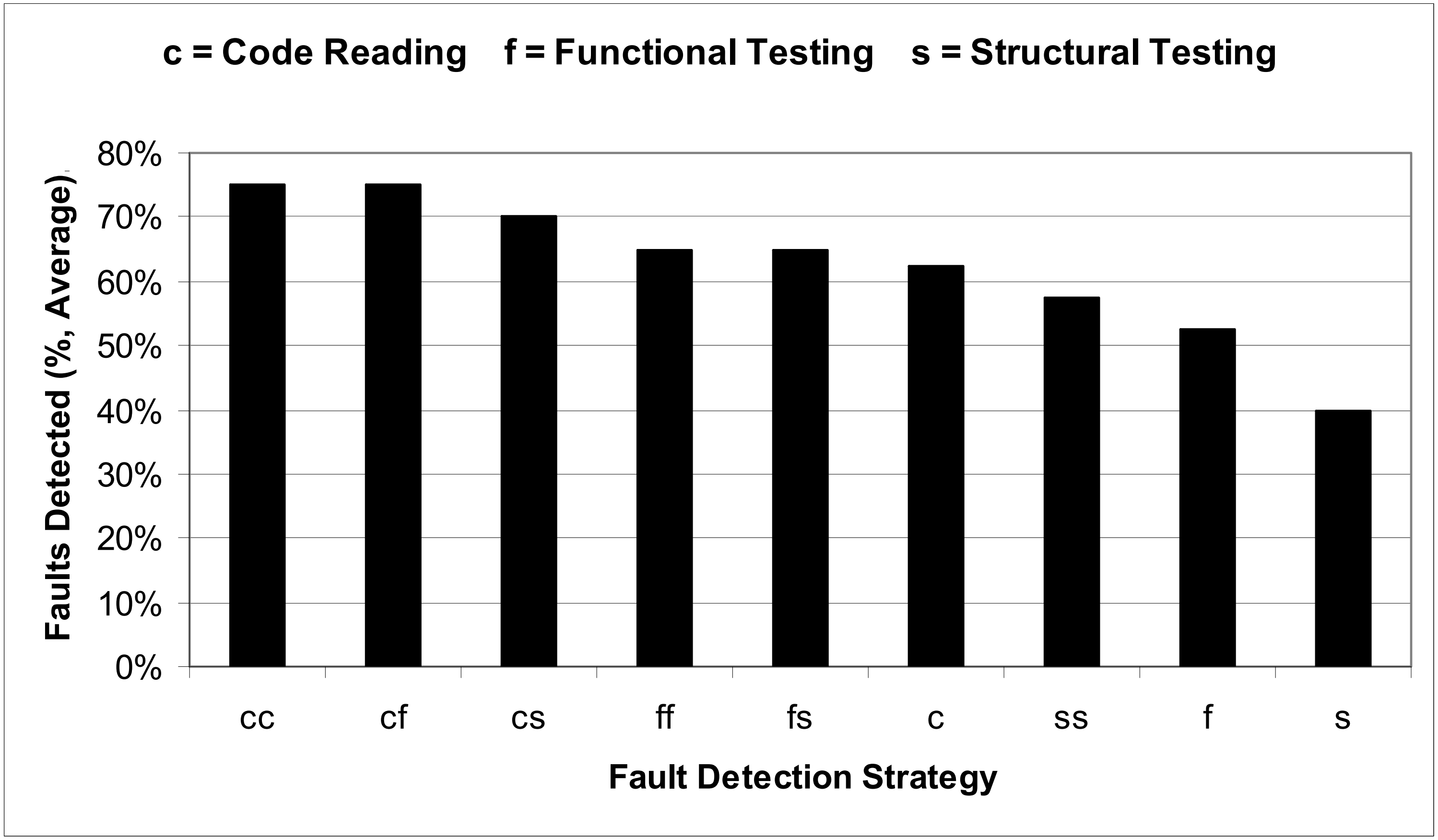
Return-on-Investment (ROI) for Software Peer Reviews Ranges from 9:1 to 3800:1 per Project



	Total	Ave.	Ave. / EKSLOC
Reviews	**257**	**29**	**N/A**
Prevention cycles	**15**	**1.7**	**N/A**
Defects	**2621**	**291**	**7.3**
Defects per review	**N/A**	**15**	**N/A**
Defects out-of-phase	**N/A**	**8.1%**	**1.3**

- **Return-on-investment (ROI) = Net cost avoidance divided by non-recurring cost**
- **2621 defects, 257 reviews, 9 systems, 1.5 years**
- **High ROI drivers**
  - **Mature and effective processes already in place**
  - **Significant new scope under development**
  - **Early lifecycle peer reviews (e.g., requirements phase)**
  - **Four of the five programs with >80% requirements and design defects had relatively higher ROI**

Analyses of Fault Detection Strategies Characterize Fault Types and Effectiveness of Teaming

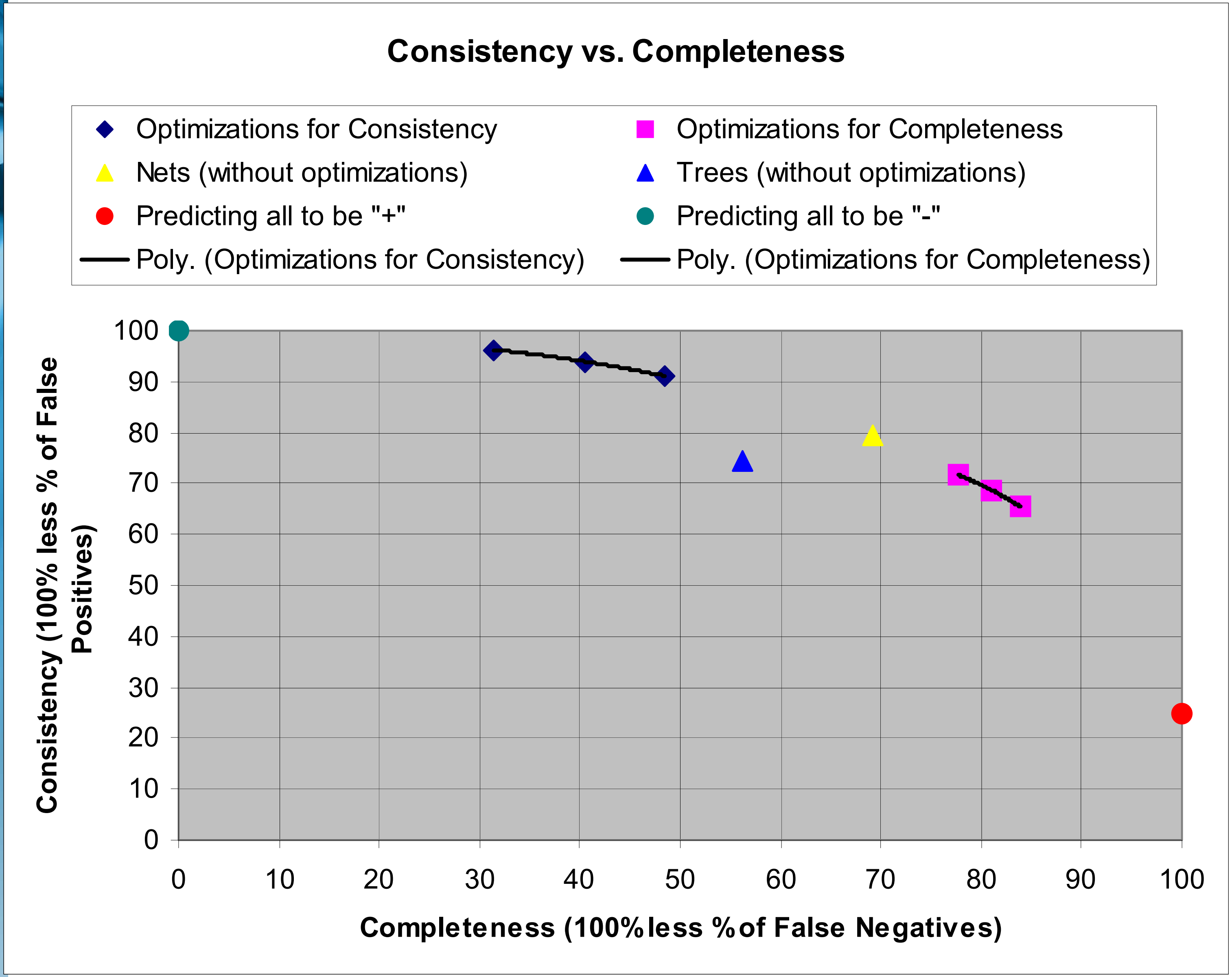


Legend

- c = Code reading by stepwise abstraction
- f = Functional testing using equivalence partitioning and boundary value analysis
- s = Structural testing using 100% statement coverage
- xy = Two-person combination of technique x and y

- Unit-level fault detection strategies for 32 NASA developers and two-person teams
- Six combined testing strategies detected 67% and three individual techniques detected 50% of the software faults on average (35% improvement)
- Highest percentage of software faults detected when there was a combination of either two code readers or a code reader and a functional tester (75%)
- Combined code reading strategies (cc/cf/cs) exceeded all individual techniques

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



- **Target: Identify error-prone (top 25%) and effort-prone (top 25%) components**
- **16 large NASA systems**
- **1920 configurations**
- **Models use metric-driven decision trees and networks**
- **Optimizations: consistency & completeness**

NORTHROP GRUMMAN

Opportunities for Improvement and Research

- Model-based engineering
- End-to-end capability analyses, tradeoff analyses, sensitivity analyses, and margin assessments
- Reuse
- Return-on-investment analyses for defining, enhancing, and pruning processes

NORTHROP GRUMMAN