# Software Metrics In Use at JPL Applications and Research

JPL
California Institute of Technology

1. **ODC-Based Anomaly Analysis**
   Dr. Robyn Lutz (JPL/ISU) & Carmen Mikulski (JPL)
2. **Assurance Optimization**
   Dr. Martin S. Feather (JPL)
3. **Predictive Cost/Quality Metrics**
   John Powell (JPL) & Dr. Jairus Hihn (JPL)
4. **Detailed Project Metrics**
   Helenann Kwong-Fu (JPL)
5. **Formal Definition of Software Faults**
   Dr. John Munson, Univ. of Idaho & Dr. Allen Nikora (JPL)


## *Presentation by Dr. Allen Nikora*

*Jet Propulsion Laboratory, California Institute of Technology*



The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology. This work is sponsored by the National Aeronautics and Space Administration's Office of Safety and Mission Assurance under the NASA Software Program led by the NASA Software IV&V Facility. This activity is managed locally at JPL through the Assurance Technology Program Office (ATPO).

# 1. ODC-Based Anomaly Analysis

JPL
California Institute of Technology

Dr. Robyn Lutz (JPL/ISU) and Carmen Mikulski (JPL)
NASA Code Q Software Program Center Initiative UPN 323-08
Supported by JPL Assurance Technology Program Office

Goal: To reduce the number of safety-critical software anomalies that occur during flight by providing a quantitative analysis of previous anomalies as a foundation for process improvement

Analyzed 199 Incident/Surprise/Anomaly reports (ISAs)

- Software anomalies post-launch
- High criticality
- 7 spacecraft
- Institutional defect database

# ODC-Based Anomaly Analysis *Approach*

JPL
California Institute of Technology

- Orthogonal Defect Classification (ODC) method
  - Developed at IBM; widely used by industry
  - Quantitative approach
  - Used here to detect patterns in anomaly data
- Adapted ODC categories to operational spacecraft software at JPL
  - Activity: what was taking place when anomaly occurred?
  - Trigger: what was the catalyst?
  - Target: what was fixed?
  - Type: what kind of fix was done?
- Collaborating with Mars Exploration Rover to experimentally extend to pre-launch testing

# ODC-Based Anomaly Analysis *Example*

JPL
California Institute of Technology

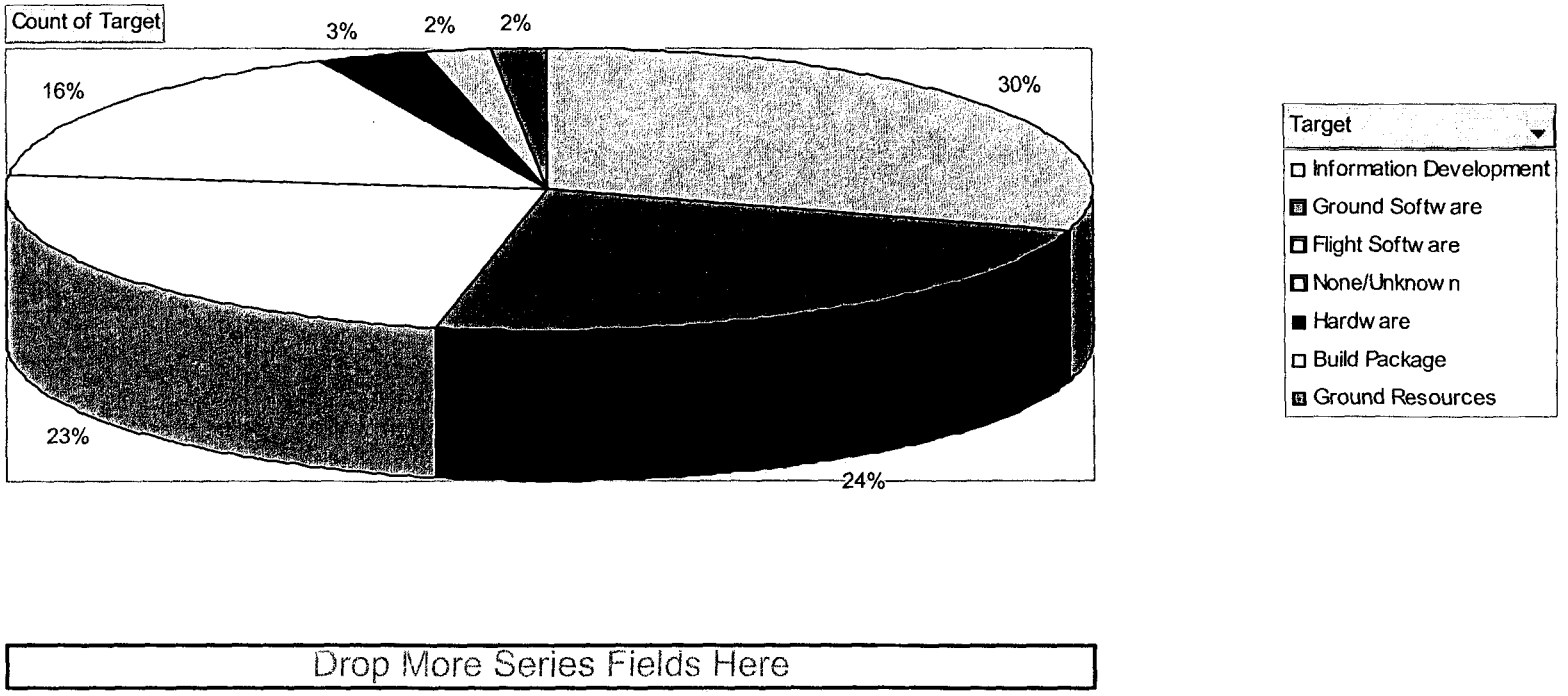
- Sample Question: What is the typical signature of a post-launch critical software anomaly?
- Metrics:
  - Activity = Flight Operations
  - Trigger = Data Access/Delivery
  - Target = Information Development
  - Type = Procedures
- Example: Star Scanner anomaly
  - Activity = occurred during flight
  - Trigger = star scanner telemetry froze
  - Target = fix was new description of star calibration
  - Type = procedure written

# ODC-Based Anomaly Analysis
## *Quantitative Analysis*

JPL
California Institute of Technology

PROJECT (All)

**Target Distribution**



# ODC-Based Anomaly Analysis
## *Benefits*

JPL
California Institute of Technology

- User selects preferred representation (e.g., 2-D bar graph) and set of projects to view
- Data mines historical and current databases of anomaly and problem reports
- Uses metrics information to identify unexpected patterns and focus on problem areas
- Provides rapid quantitative foundation for process improvement
- Equips us with a methodology to continue to learn as projects and processes evolve

# 2. Assurance Optimization

JPL
California Institute of Technology

Dr. Martin S. Feather (JPL)

NASA Code Q Software Program Center Initiative UPN 323-08
Supported by JPL Assurance Technology Program Office

Assurance activities have costs:

- Requirements inspections take skilled peoples' time
- Test-what-you-fly takes high-fidelity testbeds

Assurance activities have **benefits**:

- Requirements inspections may **catch problems early**, when it is inexpensive to fix them
- Test-what-you-fly may **catch problems that would jeopardize the mission**

# Assurance Optimization
## *Goals*

JPL
California Institute of Technology

The selection of assurance activities such that:

For a given set of resources
(time, budget, personnel, test beds, mass, power, ...)
**benefits are maximized**

or

**For a given level of benefits attainment**
(science return goals; on-time and in-budget development, ...)
costs are minimized

# Assurance Optimization
## *What's Needed to do it*

JPL
California Institute of Technology

**I** A model to calculate assurance costs & **benefits** -
we use Dr. Steve Cornford's Defect Detection and Prevention (DDP) (http://ddptool.jpl.nasa.gov)

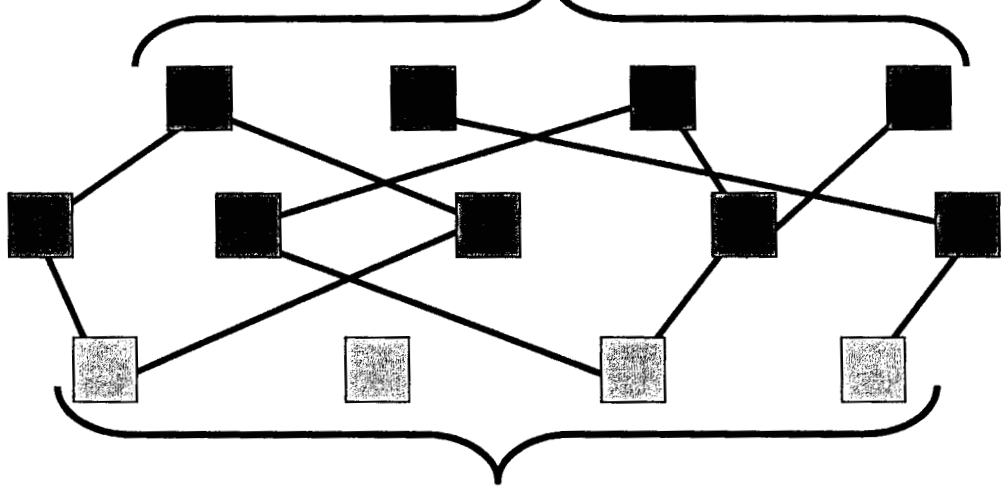
**II** Optimization over the model -
we use Menzies' TAR2 treatment learning system (http://www.tim.menzies.com)
Also exploring use of genetic algorithms.

**III** Data to populate the model -
we populate with **metrics** from experience (when available) augmented with experts' best estimates

# Assurance Optimization
## *I. DDP Cost/Benefit Model*

California Institute of Technology

**Benefits** = Σ attainment of requirements



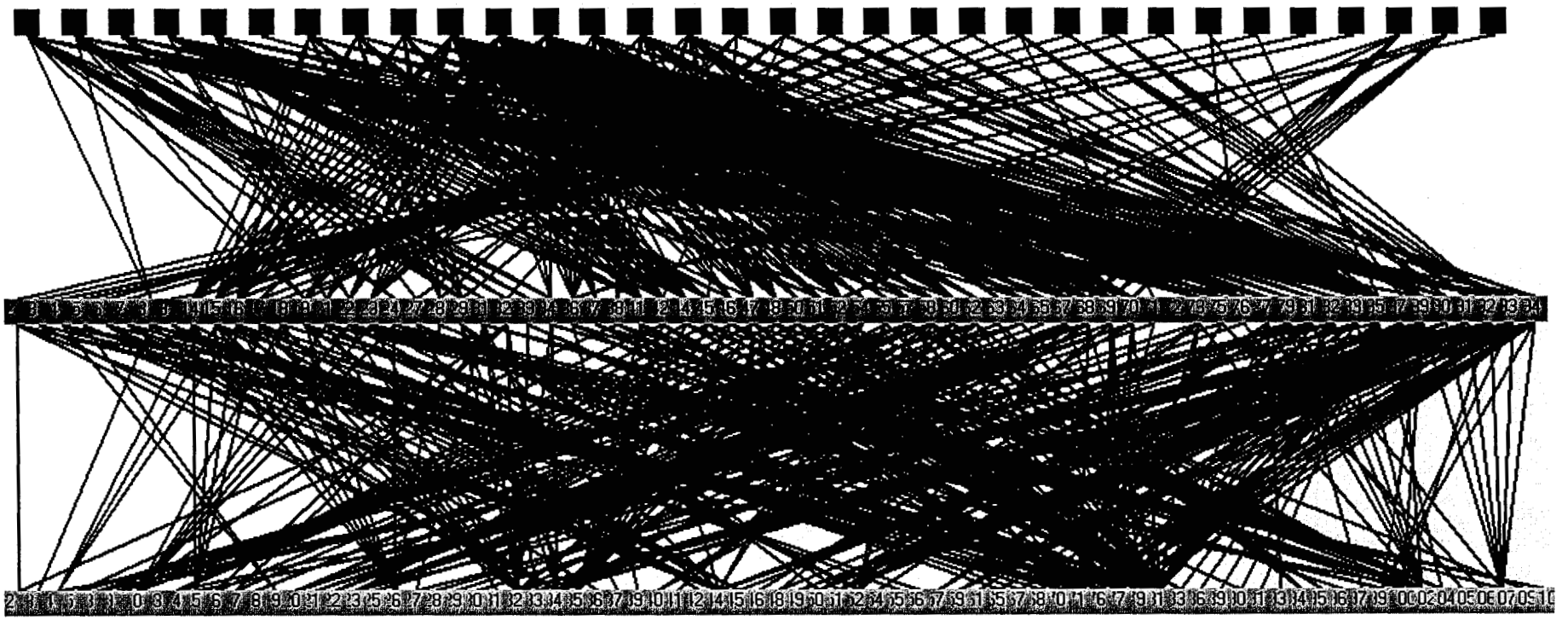

**Costs** = Σ costs of selected assurance activities

Model holds *quantitative* measures of:
*How much* each risk impacts each requirement, and
*How much* each assurance activity reduces each risk.

**Risks** are crucial intermediaries in the model -
requirements impacted by **risks** to differing extents
assurance activities mitigate **risks** to differing extents

# Assurance Optimization
## *A Populated DDP Dataset*

JPL
California Institute of Technology

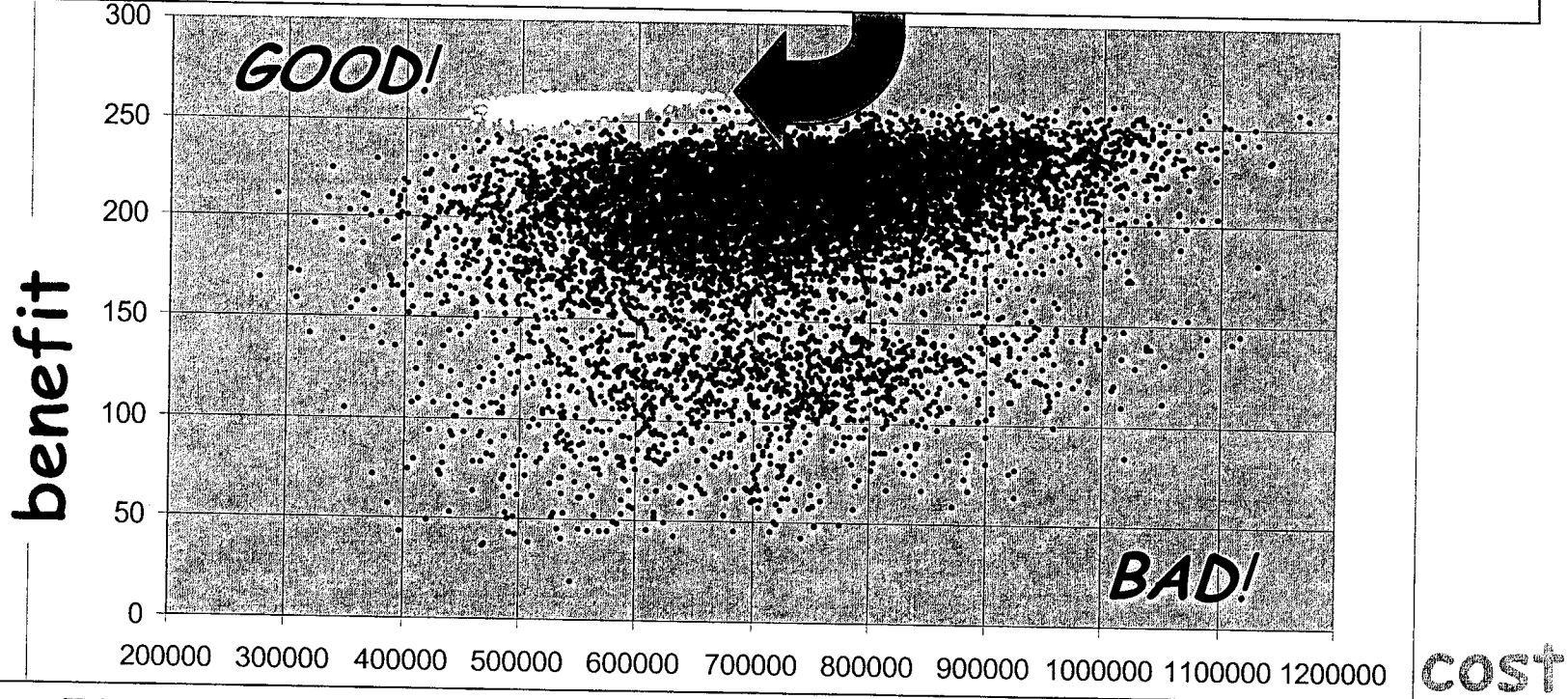


32 requirements, 69 risks, 99 assurance activities
352 non-zero quantitative requirement-risk links
440 non-zero quantitative assurance-risk links

# Assurance Optimization
## *II. Finding the Optimum*

JPL California Institute of Technology

Each black point a randomly chosen selection of dataset's assurance activities. DDP used to calculate **cost** and **benefit** of each such selection. Each white point is an *optimized* selection of dataset's assurance activities.




Tim Menzies' TAR2 system used here for optimization.
TAR2 identified 33 most critical decisions:
21 of them assurance activities to perform
12 of them assurance activities to *not* perform.

# Assurance Optimization
## *III. Data to Populate the Model*

JPL California Institute of Technology

**Risks:**

How many defects, and of what kinds, are introduced during development? How serious are those defects?

**Assurance activities:**

How effective are assurance activities at defect prevention, and at detection & repair? How much do they cost?

***Metrics** can help provide this data:*

Historical, culled from past similar cases.

Predictive, during current development.

Monitoring, to track current development against predictions.

In the absence of metrics we use expert judgment.

**We need your metrics!**

# 3. Predictive Cost/Quality Metrics

California Institute of Technology

John Powell (JPL) & Dr. Jairus Hihn (JPL)

Supported by JPL's Software Quality Improvement Project

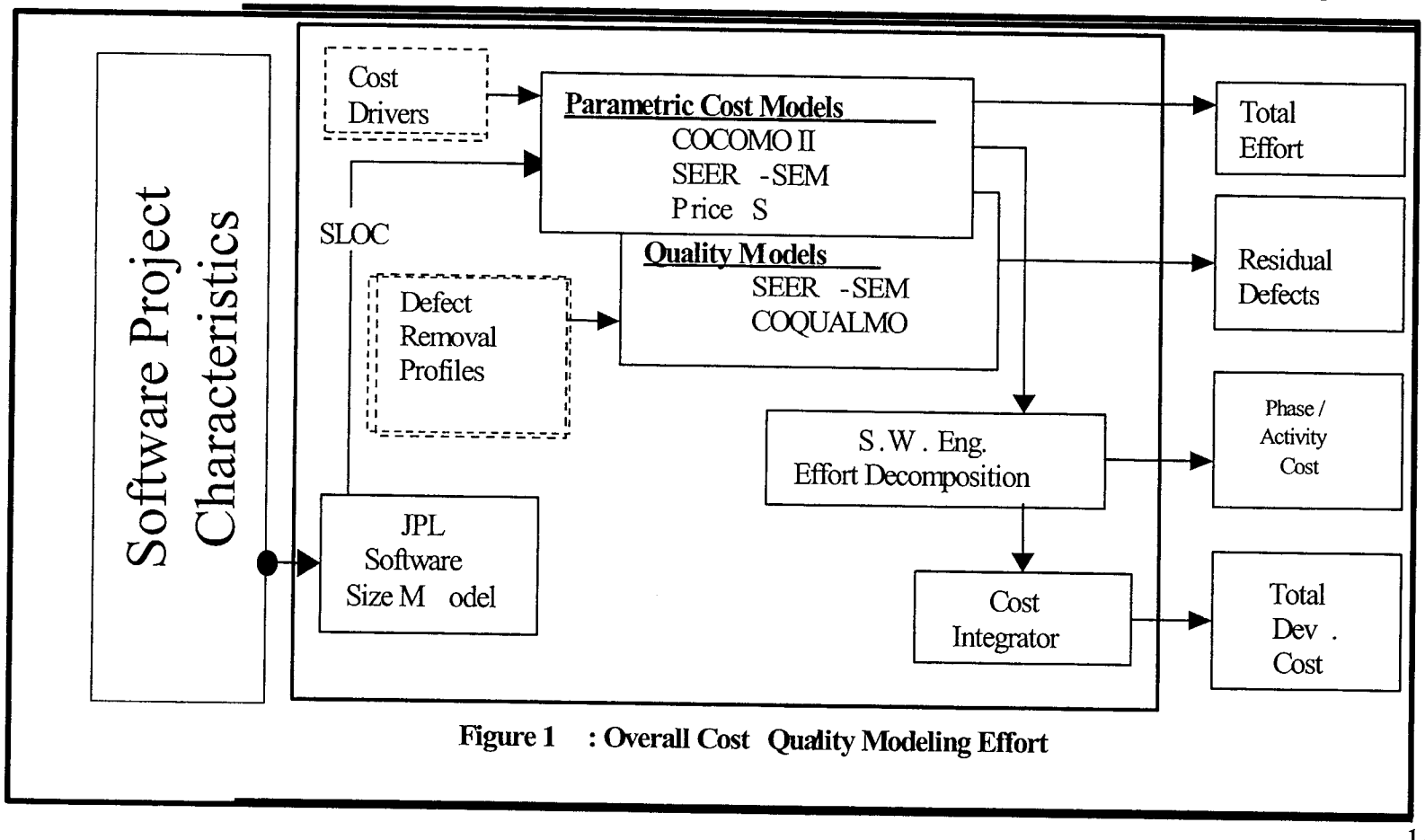



**Figure 1 : Overall Cost Quality Modeling Effort**

# Predictive Cost/Quality Metrics
## *Input Data for Cost / Quality Models*

JPL
California Institute of Technology

- **Cost Models**
  - **Size SLOC, Function Points**
  - **COCOMO II Cost Drivers and Scale Factors**
  - **Straight Forward Mapping to SEER-SEM**
- **Quality Models**
  - **Three Defect Removal Profile Levels**
  - **Cost Drivers from Cost Model Inputs**

# Predictive Cost/Quality Metrics
## *Model Outputs for Calibration*

JPL
California Institute of Technology

- **Cost Models**
  - Planned and Actual Effort
  - Suitably broken down by phase or activity or function etc… (Specific to organization needs)
- **Quality Model**
  - Tracked Software Defects
  - Categorized by pinot of introduction (Requirements, Design, Code)

# Predictive Cost/Quality Metrics
## *Cost / Quality Benefits*

JPL
California Institute of Technology

- Consider a Navigation software component
  - If budget/effort is inaccurately estimated for NAV
    - Cost over runs may disrupt work on NAV as well as other software within the system such as fault protection etc… due to unplanned staffing and resource reallocations.
    - Budgetary Constraints due to overuns may cause project cancellation.
  - If defect density is inaccurately estimated
    - Insufficient or unnecessary resources may be allocated for QA, V&V, Testing etc… of navigation (or other) system software
    - Unexpected delays may occur due to larger than expected volume of defect repairs or late discovery of defects may jeopardize budget, schedule, delivery date and/or system functionality

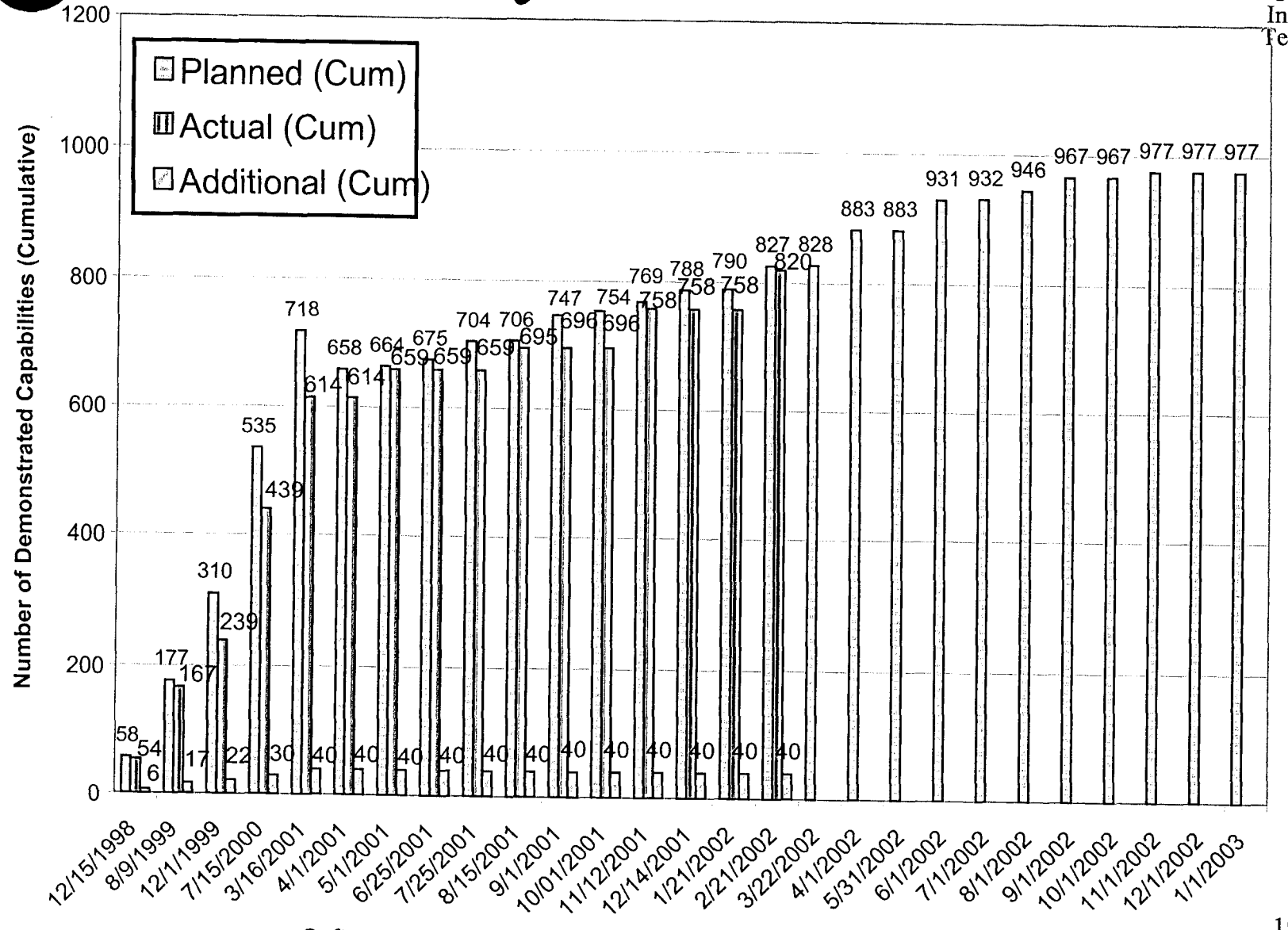
# 4. Detailed Project Metrics

JPL
California Institute of Technology

## Helenann Kwong-Fu (JPL)
## JPL Software Quality Assurance

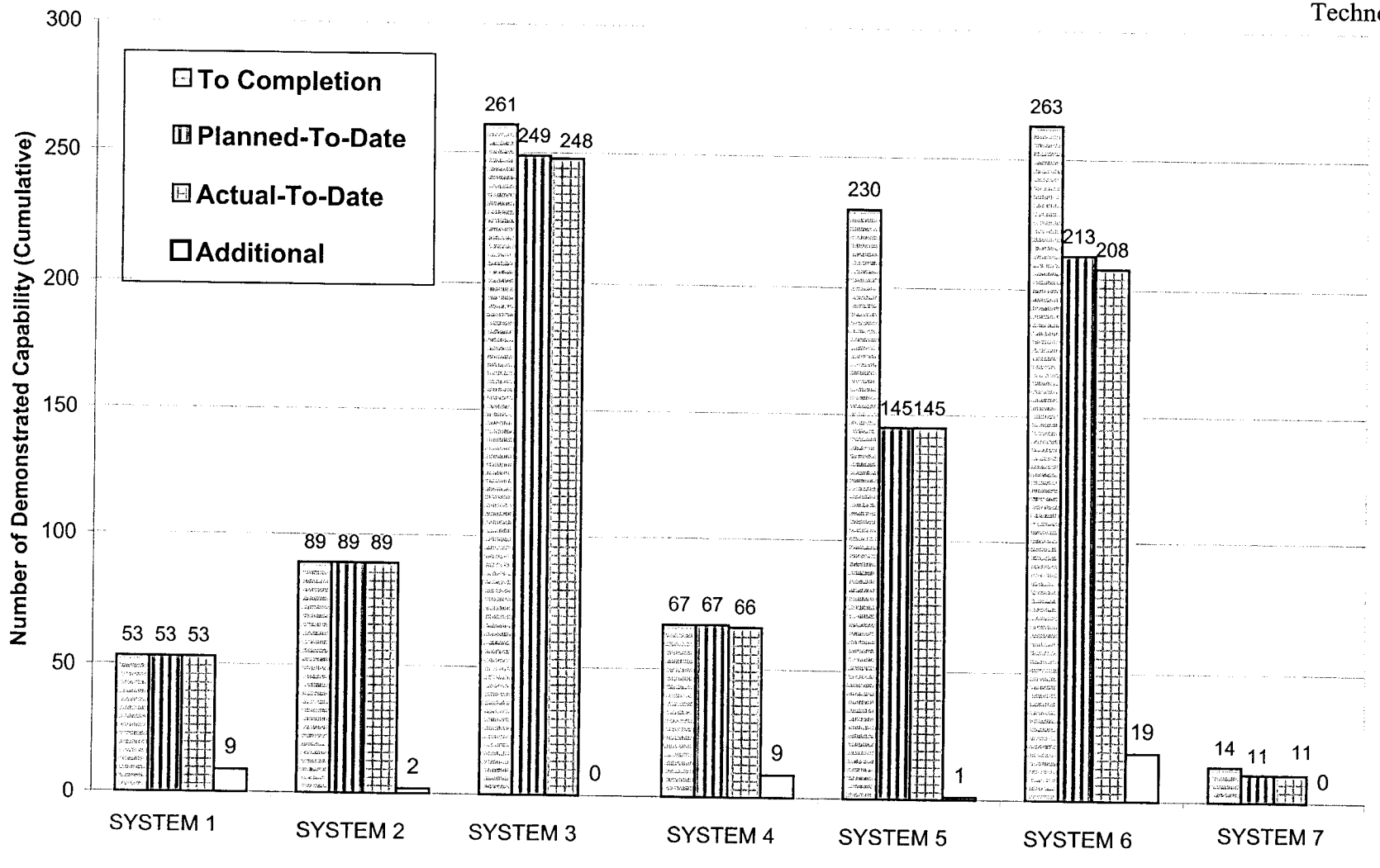
*A sampling of actual SQA metrics kept on a project*

# Detailed Project Metrics
## *Overall Project Software Metrics*



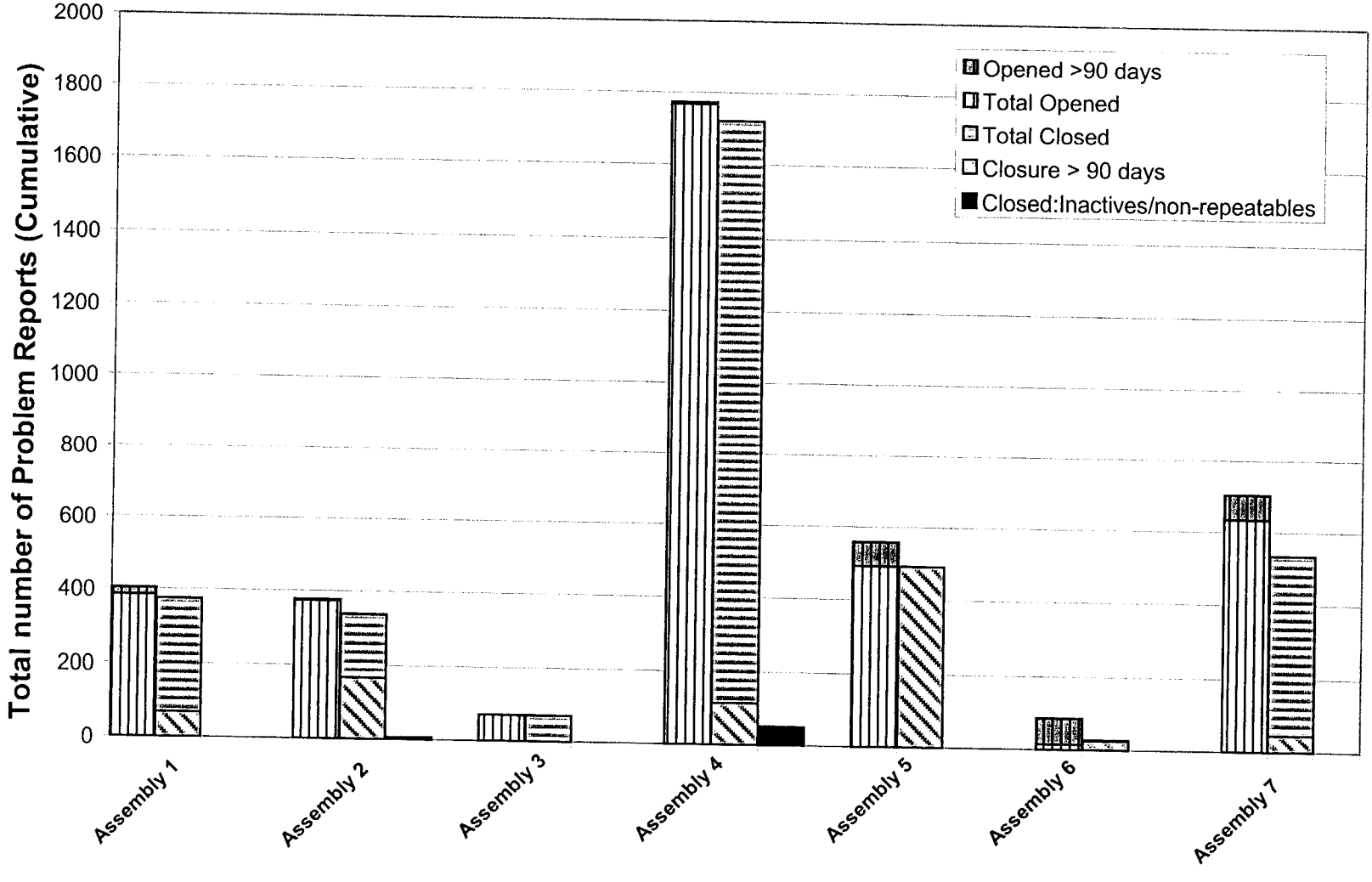

Software Metrics In Use at JPL-Applications and Research

# Detailed Project Metrics
## *Software Metrics By System*



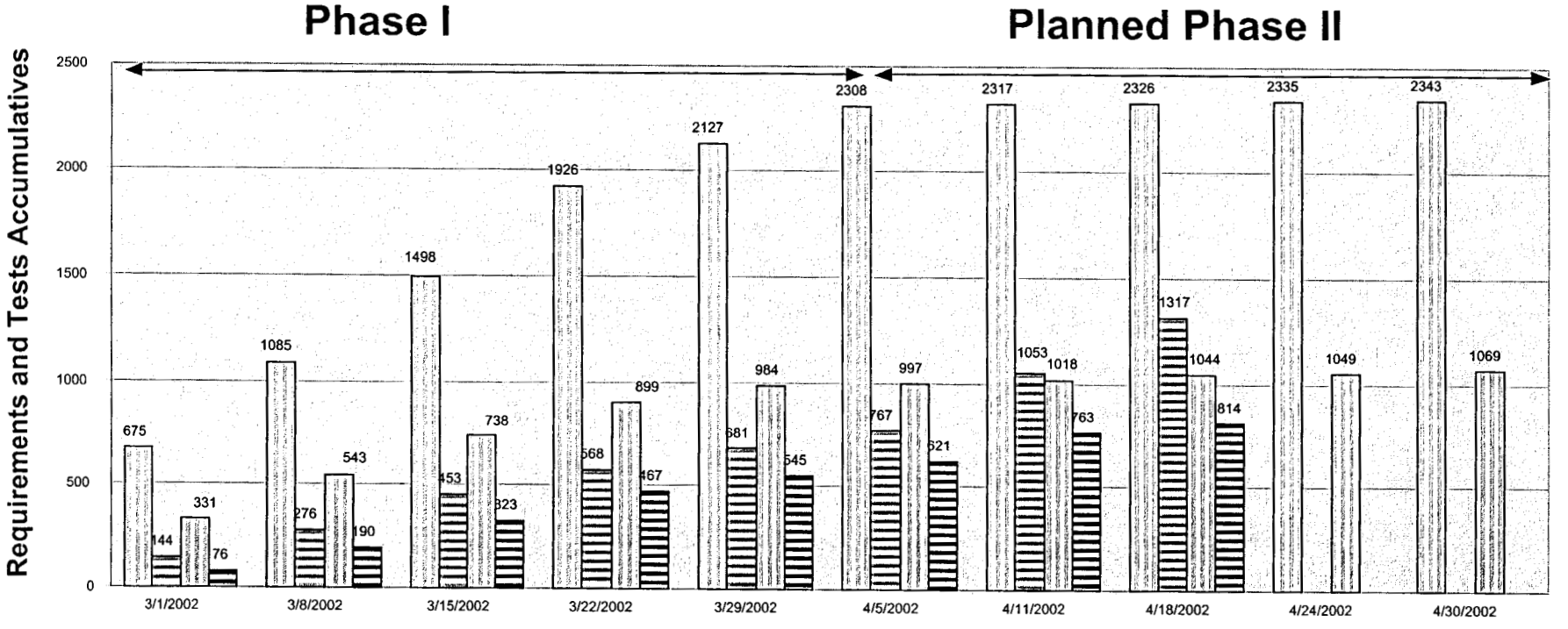
# Detailed Project Metrics
## *Project Integration & Test Problem Reports*

JPL California Institute of Technology



# Detailed Project Metrics
## *Completion of Requirements and Tests*

California Institute of Technology




#Req. Planned to complete

# Req. Passed (Actuals)

#Test Planned to complete

Total test/SE signed off (Actuals)

# Detailed Project Metrics
## *Implementation Process Risk Metrics*
## *Team-by-Team*

California Institute of Technology

**TEAM 1**

87% AI 7_00

| Obj. | Priority | Risk |
|---|---|---|
| 11.1 | 3 | 4 |
| 11.2 | 3 | 4 |
| 11.3 | 3 | 4 |
| 3.4 | 2 | 2 |
| 4.6 | 2 | 2 |
| 4.8 | 1 | 2 |
| 5.2 | 1 | 2 |
| 9.3 | 1 | 2 |
| 1.3 | 3 | 1 |
| 1.4 | 3 | 1 |
| 1.5 | 3 | 1 |
| 2.1 | 3 | 1 |
| 2.3 | 3 | 1 |
| 2.4 | 3 | 1 |
| 4.2 | 3 | 1 |
| 4.3 | 3 | 1 |
| 6.2 | 3 | 1 |
| 8.1 | 3 | 1 |
| 8.2 | 3 | 1 |
| 9.2 | 3 | 1 |
| 2.2 | 2 | 1 |
| 3.2 | 2 | 1 |
| 3.3 | 2 | 1 |
| 3.5 | 2 | 1 |
| 5.1 | 1 | 1 |
| 7.1 | 1 | 1 |
| 9.1 | 1 | 1 |
| 2.6 | 3 | 0 |
| 4.4 | 3 | 0 |
| 6.1 | 3 | 0 |
| 2.5 | 2 | 0 |
| 3.1 | 2 | 0 |

**TEAM 2**

46.7% AI

| Obj. | Priority | Risk |
|---|---|---|
| 12 | 2.3 | 3 | 4 |
| 12 | 6.1 | 3 | 4 |
| 12 | 6.2 | 3 | 4 |
| 9 | 2.5 | 2 | 4 |
| 9 | 4.6 | 2 | 4 |
| 3 | 5.1 | 1 | 2 |
| 3 | 9.1 | 1 | 2 |
| 3 | 9.3 | 1 | 2 |
| 3 | 1.4 | 3 | 1 |
| 3 | 2.1 | 3 | 1 |
| 3 | 2.4 | 3 | 1 |
| 3 | 4.5 | 3 | 1 |
| 3 | 2.2 | 2 | 1 |
| 3 | 3.1 | 2 | 1 |
| 3 | 3.3 | 2 | 1 |
| 3 | 3.4 | 2 | 1 |
| 2 | 4.4 | 3 | 0 |
| 2 | 3.2 | 2 | 0 |
| 2 | 10.1 | 3 | -1 |
| 2 | 10.2 | 3 | -1 |
| 1 | 10.3 | 3 | -1 |
| 1 | 10.4 | 3 | -1 |
| 1 | 10.5 | 3 | -1 |
| 0 | 10.6 | 3 | -1 |
| 0 | 10.7 | 3 | -1 |
| 0 | 11.1 | 3 | -1 |
| 0 | 11.2 | 3 | -1 |
| 0 | 11.3 | 3 | -1 |

**TEAM 3**

74.3% AI

| Obj. | Priority | Risk |
|---|---|---|
| 12 | 4.1 | 3 | 4 |
| 12 | 4.2 | 3 | 4 |
| 12 | 4.3 | 3 | 4 |
| 8 | 4.4 | 3 | 4 |
| 8 | 4.5 | 3 | 4 |
| 4 | 4.7 | 3 | 4 |
| 4 | 6.2 | 3 | 4 |
| 4 | 11.1 | 3 | 4 |
| 6 | 11.2 | 3 | 4 |
| 6 | 11.3 | 3 | 4 |
| 6 | 3.5 | 2 | 4 |
| 6 | 4.6 | 2 | 4 |
| 2 | 2.2 | 2 | 2 |
| 2 | 2.5 | 2 | 2 |
| 3 | 3.1 | 2 | 2 |
| 3 | 5.1 | 1 | 2 |
| 3 | 9.1 | 1 | 2 |
| 3 | 9.3 | 1 | 2 |
| 2 | 1.1 | 3 | 1 |
| 2 | 1.2 | 3 | 1 |
| 2 | 1.4 | 3 | 1 |
| 2 | 1.5 | 3 | 1 |
| 0 | 2.4 | 3 | 1 |
| 0 | 10.1 | 3 | 1 |
| -3 | 10.2 | 3 | 1 |
| -3 | 10.3 | 3 | 1 |
| -3 | 3.2 | 2 | 1 |
| -3 | 3.3 | 2 | 1 |
| -3 | 3.4 | 2 | 1 |
| -3 | 7.1 | 1 | 1 |
| -3 | 10.4 | 3 | 0 |
| -3 | 10.5 | 3 | 0 |
| -3 | 10.6 | 3 | 0 |
| -3 | 10.7 | 3 | 0 |

**TEAM 4**

88.0% AI 6_99

| Obj. | Priority | Risk |
|---|---|---|
| 12 | 4.7 | 3 | 4 |
| 12 | 11.1 | 3 | 4 |
| 12 | 11.2 | 3 | 4 |
| 12 | 11.3 | 3 | 4 |
| 12 | 5.1 | 1 | 2 |
| 12 | 7.1 | 1 | 2 |
| 12 | 1.2 | 3 | 1 |
| 8 | 1.3 | 3 | 1 |
| 8 | 1.5 | 3 | 1 |
| 4 | 2.1 | 3 | 1 |
| 4 | 2.3 | 3 | 1 |
| 9 | 2.6 | 3 | 1 |
| 6 | 4.1 | 3 | 1 |
| 6 | 4.3 | 3 | 1 |
| 6 | 4.4 | 3 | 1 |
| 6 | 6.1 | 3 | 1 |
| 6 | 8.1 | 3 | 1 |
| 6 | 9.2 | 3 | 1 |
| 6 | 3.2 | 2 | 1 |
| 4 | 3.4 | 2 | 1 |
| 4 | 3.5 | 2 | 1 |
| 4 | 4.6 | 2 | 1 |
| 2 | 4.8 | 1 | 1 |
| 2 | 1.1 | 3 | 0 |
| 2 | 1.4 | 3 | 0 |
| 3 | 6.2 | 3 | 0 |
| 3 | 8.2 | 3 | 0 |
| 3 | 10.1 | 3 | 0 |
| 3 | 10.2 | 3 | 0 |
| 3 | 10.3 | 3 | 0 |
| 3 | 10.4 | 3 | 0 |
| 3 | 10.5 | 3 | 0 |
| 3 | 10.6 | 3 | 0 |
| 2 | 10.7 | 3 | 0 |
| 2 | 2.2 | 2 | 0 |
| 2 | 2.5 | 2 | 0 |
| 1 | 3.1 | 2 | 0 |
| 0 | 3.3 | 2 | 0 |
| 0 | 5.2 | 1 | 0 |
| 0 | 9.1 | 1 | 0 |
| 0 | 9.3 | 1 | 0 |

As of 6/1999

| Color | Definitions |
|---|---|
| | No Information |
| | Highest Concerns |
| | Sig. Concerns |
| | Least-No concerns |

| Obj.# | Definitions |
|---|---|
| 1.X | S/W Management |
| 2.X | S/W Requirement and Design |
| 3.X | S/W Dev. Std./Coding Conv. & Maint. |
| 4.X | S/W Test Verification and Validation |
| 5.X | S/W Development Tools |
| 6.X | S/W Problem Reporting/Resolution |
| 7.X | S/W Documentation |
| 8.X | S/W Configuration Management |
| 9.X | S/W Quality Control |
| 10.X | S/W Fault Protection |
| 11.X | S/W Safety |

| Ref. | Risk | Priority |
|---|---|---|
| -1 | N/A | — |
| 0 | none | — |
| 1 | low | low |
| 2 | Sig | Med |
| 3 | High | High |
| 4 | Pend | — |

# Detailed Project Metrics
## *Implementation Process Risk Metrics*
## *One Team's Progress Over Time*

California Institute of Technology

**JUN 99**

87% AI — Jun-99

| Obj. | Priority | Risk | |
|---|---|---|---|
| 1.1 | 3 | 1 | |
| 1.2 | 3 | 1 | |
| 1.3 | 3 | 0 | |
| 1.4 | 3 | 1 | |
| 1.5 | 3 | 1 | |
| 2.1 | 3 | 1 | |
| 2.2 | 2 | 1 | |
| 2.3 | 3 | 1 | |
| 2.4 | | | |
| 2.5 | 2 | 0 | |
| 2.6 | 3 | 0 | |
| 3.1 | 2 | 0 | |
| 3.2 | 2 | 1 | |
| 3.3 | 2 | 1 | |
| 3.4 | 2 | 1 | |
| 3.5 | 2 | 0 | |
| 4.1 | | | |
| 4.2 | 3 | 0 | |
| 4.3 | 3 | 1 | |
| 4.4 | 3 | 0 | |
| 4.5 | | | |
| 4.6 | 2 | 2 | |
| 4.7 | 3 | 4 | |
| 4.8 | 1 | 2 | |
| 5.1 | 1 | 1 | |
| 5.2 | 1 | 2 | |
| 6.1 | 3 | 1 | |
| 6.2 | 3 | 1 | |
| 7.1 | 1 | 1 | |
| 8.1 | | | |
| 8.2 | 3 | 1 | |
| 9.1 | 1 | 1 | |
| 9.2 | | | |
| 9.3 | 1 | 1 | |
| 10.1 | | | |
| 10.2 | | | |
| 10.3 | | | |
| 10.4 | | | |
| 10.5 | | | |
| 10.6 | | | |
| 10.7 | | | |
| 11.1 | 3 | 4 | |
| 11.2 | 3 | 4 | |
| 11.3 | 3 | 4 | |

**JAN 00**

87% AI — Jan-00

| Obj. | Priority | Risk | |
|---|---|---|---|
| 1.1 | | | |
| 1.2 | | | |
| 1.3 | | | |
| 1.4 | 3 | 1 | |
| 1.5 | | | |
| 2.1 | | | |
| 2.2 | 2 | 1 | * |
| 2.3 | | | |
| 2.4 | 3 | 1 | |
| 2.5 | 2 | 0 | * |
| 2.6 | 3 | 0 | |
| 3.1 | 2 | 1 | |
| 3.2 | 2 | 1 | * |
| 3.3 | 2 | 1 | * |
| 3.4 | 2 | 2 | |
| 3.5 | 2 | 2 | |
| 4.1 | | | |
| 4.2 | | | |
| 4.3 | 3 | 1 | * |
| 4.4 | 3 | 0 | * |
| 4.5 | | | |
| 4.6 | 2 | 2 | |
| 4.7 | | | |
| 4.8 | 1 | 2 | * |
| 5.1 | 1 | 1 | |
| 5.2 | 1 | 2 | * |
| 6.1 | 3 | 1 | |
| 6.2 | 3 | 1 | |
| 7.1 | 1 | 2 | |
| 8.1 | | | |
| 8.2 | | | |
| 9.1 | 1 | 1 | * |
| 9.2 | | | |
| 9.3 | | | |
| 10.1 | | | |
| 10.2 | | | |
| 10.3 | | | |
| 10.4 | | | |
| 10.5 | | | |
| 10.6 | | | |
| 10.7 | | | |
| 11.1 | 3 | 4 | |
| 11.2 | 3 | 4 | |
| 11.3 | 3 | 4 | |

**JUL 00**

87% AI — Jul-00

| Obj. | Priority | Risk | |
|---|---|---|---|
| 1.1 | | | |
| 1.2 | | | |
| 1.3 | 3 | 1 | |
| 1.4 | 3 | 1 | |
| 1.5 | 3 | 1 | |
| 2.1 | 3 | 1 | |
| 2.2 | 2 | 1 | * |
| 2.3 | 3 | 1 | |
| 2.4 | 3 | 1 | |
| 2.5 | 2 | 0 | * |
| 2.6 | 3 | 0 | |
| 3.1 | 2 | 0 | |
| 3.2 | 2 | 1 | * |
| 3.3 | 2 | 1 | * |
| 3.4 | 2 | 2 | |
| 3.5 | 2 | 1 | |
| 4.1 | | | |
| 4.2 | 3 | 1 | |
| 4.3 | 3 | 1 | * |
| 4.4 | 3 | 0 | * |
| 4.5 | | | |
| 4.6 | 2 | 2 | |
| 4.7 | | | |
| 4.8 | 1 | 2 | * |
| 5.1 | 1 | 1 | |
| 5.2 | 1 | 2 | * |
| 6.1 | 3 | 0 | |
| 6.2 | 3 | 1 | |
| 7.1 | 1 | 1 | |
| 8.1 | 3 | 1 | |
| 8.2 | 3 | 1 | |
| 9.1 | 1 | 1 | * |
| 9.2 | 3 | 1 | |
| 9.3 | 1 | 2 | |
| 10.1 | | | |
| 10.2 | | | |
| 10.3 | | | |
| 10.4 | | | |
| 10.5 | | | |
| 10.6 | | | |
| 10.7 | | | |
| 11.1 | 3 | 4 | |
| 11.2 | 3 | 4 | |
| 11.3 | 3 | 4 | |

As of 6/1999

| Color | Definitions |
|---|---|
| | No Information |
| | Highest Concerns |
| | Sig. Concerns |
| | Least-No concerns |

| Obj.# | Definitions |
|---|---|
| 1.X | S/W Management |
| 2.X | S/W Requirement and Design |
| 3.X | S/W Dev. Std./Coding Conv. & Maint. |
| 4.X | S/W Test Verification and Validation |
| 5.X | S/W Development Tools |
| 6.X | S/W Problem Reporting/Resolution |
| 7.X | S/W Documentation |
| 8.X | S/W Configuration Management |
| 9.X | S/W Quality Control |
| 10.X | S/W Fault Protection |
| 11.X | S/W Safety |

| Ref. | Risk | Priority |
|---|---|---|
| -1 | N/A | — |
| o | none | — |
| 1 | low | low |
| 2 | Sig | Med |
| 3 | High | High |
| 4 | Pend | — |

# 5. Formal Definition of Software Faults

JPL
California Institute of Technology

- **Investigators**
  - **Dr. John Munson, University of Idaho**
  - **Dr. Allen Nikora, JPL**

**NASA Code Q Software Program Center Initiative UPN 323-08**
**Supported by JPL Assurance Technology Program Office**

# Formal Definition of Software Faults
## *Motivation*

JPL California Institute of Technology

- Developing software fault models depends on definition of what constitutes a fault
- Desired characteristics of measurements, measurement process
  - Repeatable, accurate count of faults
  - Measure at same level at which structural measurements are taken
    - Measure at module level (e.g., function, method)
  - Easily automated

# Formal Definition of Software Faults
## *Current State of Affairs*

JPL California Institute of Technology

- No existing definition of fault in measurable terms
  - IEEE Standards
    - IEEE Std 729-1983, "IEEE Standard Glossary of Software Engineering Terminology"
    - IEEE Std 982.1-1988, "IEEE Standard Dictionary of Measures to Produce Reliable Software"
    - IEEE Std 1044-1993, "IEEE Standard Classification for Software Anomalies"
  - ODC
  - Previous work (Annual Oregon Workshop on Software Metrics, May 11-13, 1997)
  - Frankl, Hamlet, Littlewood, Stringini (IEEE TSE, vol. 24, no. 8, August 1998)

# Formal Definition of Software Faults *Approach*

JPL California Institute of Technology

- Examine changes made in response to reported failures
- Base recognition/enumeration of software faults on the grammar of the software system's language
  - Faults found in executable, non-executable statements
- Fault measurement granularity in terms of tokens that have changed

# Formal Definition of Software Faults *Approach (continued)*

JPL California Institute of Technology

- Example 1
  - Original statement: a = b + c * d;
  - Intended statement: a = b + c / d;
  - One token changed – "*" ⇒ "/"
    - Coding error
  - Count number of faults as 1

# Formal Definition of Software Faults *Approach (continued)*

JPL California Institute of Technology

- **Example 2**
  - **Original statement: a = b + c * d;**
  - **Intended statement: a = b + (c * x) + sin(z);**
  - **Substantial difference between first and second statements**
    - **Reflects design rather than coding problem**
  - **Fault measurement method should reflect the degree of change**

# Formal Definition of Software Faults *Approach (continued)*

JPL California Institute of Technology

- Consider each line of text in each version of the program as a bag of tokens
  - If a change spans multiple lines of code, all lines for the change are included in the same bag
- Number of faults based on bag differences between
  - Version of program exhibiting failures
  - Version of program modified in response to failures
- Use version control system to distinguish between
  - Changes due to repair and
  - Changes due to functionality enhancements and other non-repair changes

# Formal Definition of Software Faults
## *Approach (continued)*

JPL California Institute of Technology

- Example 1
  - Original statement: a = b + c;
    - $B_1$ = {<a>, <=>, <b>, <+>, <c>}
  - Modified statement: a = b - c;
    - $B_2$ = {<a>, <=>, <b>, <->, <c>}
  - $B_1 - B_2$ = {<+>, <-> }
  - $|B_1| = |B_2|$, $|B_1 - B_2| = 2$
  - One token has changed $\Rightarrow$ 1 fault

# Formal Definition of Software Faults *Approach (continued)*



- Example 2
  - Original statement: a = b - c;
    - $B_2$ = {<a>, <=>, <b>, <->, <c>}
  - Modified statement: a = c - b;
    - $B_3$ = {<a>, <=>, <c>, <->, <b>}
  - $B_2 - B_3 = \{\}$
  - $|B_2| = |B_3|$ , $|B_2 - B_3| = 0$
  - 1 fault representing incorrect sequencing

# Formal Definition of Software Faults *Approach (continued)*

JPL California Institute of Technology

- Approach (cont'd)
  - Example 3
    - Original statement: a = b - c;
      - $B_3$ = {<a>, <=>, <c>, <->, <b>}
    - Modified statement: a = 1 + c - b;
      - $B_4$ = {<a>, <=>, <1>, <+>, <c>, <->, <b>}
    - $B_3 - B_4$ = {<1>, <+>}
    - $|B_3| = 6$ , $|B_4| = 8$ , $|B_4| - |B_3| = 2$
    - 2 new tokens representing 2 faults

# Formal Definition of Software Faults
## *Current Work*

JPL California Institute of Technology

- Application to JPL software development effort
  - Research
  - Production
- Develop better models relating
  - Structural measurements of software evolution during development
  - Number and types of faults inserted