



Software Engineering Research Infusion

Tom Pressburger, Research Infusion Lead (ARC)
Ben Di Vito (LaRC), Martin Feather (JPL),
Michael Hinchey (GSFC), Lawrence Markosian (ARC),
Luis Trevino (MSFC)

3/23/2005



Outline

- Background
- Selected software engineering technologies



Background

- NASA Software Engineering Initiative
 - Led by the Office of the Chief Engineer
 - Improve software engineering to meet the challenges of NASA
 - Some of the areas of activity
 - Improving software development process
 - Establishing Metrics collection and analysis
 - Training the workforce
 - Improving NASA Procedures, Procedural Requirements, Standards, Directives,...
 - and....



Infusing Software Engineering Research

- Goal: Transfer into practice
 - NASA-sponsored Software Engineering Research
 - Other new software engineering tools and technologies

- Approach
 - Present **selected technologies** to the NASA software development community, and
 - Encourage and support **collaborations** between the **technology providers** and NASA **software developers**.





Collaborations

- Initiated by a **software developer** who wants to **bring on board** one or more of the technologies
- Purpose
 - benefit the software development project**
 - validate the research
 - Not:** further develop the research
- Funding available for—
 - training and consulting in the use of the technology
 - license fees in the case of commercial technologies
 - managing & applying the technology
 - collecting & analyzing data
 - reporting results.





Funding for Collaborations

- Funding for several small collaborations available from OSMA via the Software Assurance Research Program (SARP).
 - History: 10 projects in the range \$15K - \$45K: 6 in 2004, 4 in 2005
 - Competition for SARP funds is among the NASA Centers and JPL. Proposals **must** come from a **civil servant** or **JPL employee** referred to as Government Point Of Contact (POC)
 - The Principal Investigator (PI) represents the organization which actually wants to use the new technology. PI can be a **contractor** who has a contractual vehicle in place with NASA.
 - Often the POC is the COTR or technical manager on the PI's contract
 - Money is sent to the POC to put on contract
 - Either the PI or the POC can pay the technology provider.
 - Proposal template and instructions on the Research Infusion website.
 - Due: Friday, May 27, 2005, 9 AM PST.
 - Start: February 2006.
- We will help facilitate unfunded collaborations.



Software Engineering Research Infusion Sites and Applications

Technologies and Infusion sites:

C code analysis (**ARC, MSFC, IVVF**)

Formal inspection technique (**GSFC, USA**)

Defect classification (**JPL**)

Requirements analysis tool (**ARC**)

UML checking (**GSFC**)

Applications:

ISS payload, Shuttle

Spacecraft FSW, ISS

Ground system

ISS payload

Spacecraft instrument module





Success of 2004 Infusion Collaborations

- Technologies applied in 2004 Collaborations:
 - Source code analyzers
 - Formal inspection technique
 - Defect classification technique for process improvement
- Applications
 - Station and shuttle code, flight software, ground software
 - Technologies were successfully applied
 - Found defects that had escaped testing and previous inspections
 - Suggested maintenance direction
 - **Technologies were well received and most will continue to be used.**
- We are offering some of the same *as well as* new technologies for collaborations in 2006.



Training in Modern Formal Inspections

- Have proven effective time and again.
- Perspective-based approach improves efficiency; successfully applied in 2004 at GSFC and USA.
- Course developed in conjunction with Fraunhofer Institute; Dr. Forrest Shull instructor.
- 1.5 days, some pre and post work. Tailored towards your application!
- With enough interest, NASA Engineering Training (NET) will offer the course *this year*.
- If interested, email
 - Darrell.J.Thomas@nasa.gov, lead Training Subgroup of SWG.



Selected Technologies

- Culled from
 - NASA-sponsored software engineering research
 - Leading edge commercial tools
 - Research in DoD, institutes and industry suggested by NASA projects.
- Reviewed by researchers at several centers experienced in tech transfer of software engineering research.
- *Send us suggestions for next time.*
 - SE development problem areas
 - SE technologies





Selected Technologies (continued)

- Technology Selection Criteria
 - Has been successfully applied, often in a NASA context.
 - Easily adopted.
 - Focus on Software Assurance.



Collaboration Roles

For Technology Provider and Software Development Team

- Technology provider
 - During proposal preparation: help plan collaboration, including help select suitable application
 - 1 – 3 day training course at your site
 - Online tutorial and other user documentation
 - Customer support throughout collaboration



Collaboration Roles (continued)

■ Development team

- During proposal preparation:
 - Contact Research Infusion team, let us know you plan to submit proposal.
 - We will review concept and give feedback on proposal drafts!
 - Work with technology provider to plan collaboration and select suitable application.
 - Write and submit the proposal.
- Take training course.
- Identify software artifacts to which the technology will be applied.
- Apply the technology, sometimes in multiple iterations.
- Collect data & evaluate performance; write final report.



Next Step

- If you're interested in a collaboration involving one of the selected technologies, follow the proposal process at <http://ic.arc.nasa.gov/researchinfusion/>
- We want to provide feedback on proposals before the due date.



Selected Technologies

Lawrence Markosian

3/23/2005



Selected Software Engineering Research Technologies

1. Requirements Specification and Analysis
2. Matlab Testing Tools
3. Software Architecture Evaluation
4. Source Code Analysis & Error Detection
5. Linux-based RTOS
6. Software Reliability Estimation
7. Software Process Improvement





Choosing the Right Technology

- In most cases, a good choice within a group of technologies will be apparent
 - Your application
 - Technology capabilities & limitations
- Resources on our web site:
 - 1-page overview
 - 3-page technology description
 - Link to technology web site & other materials
- Talk with us to obtain a better sense of the “match”
- Narrow your choices and contact the technology vendors!





Technology Description Format

- Technology name, vendor, contact point, and NASA funding program if any
- What is it
- Features
- Benefits
- Successes
- Contexts for best use



1. Requirements Specification and Analysis

- SpecTRM
 - State-machine based requirements specification environment with particular emphasis on mission-critical and safety-critical systems

SpecTRM (Specification Tools and Requirements Methodology)



PoC: Grady Lee, Safeware Engineering
sales@safeware-eng.com, (206) 328-4880

NASA Funding for underlying research

■ What is it

- Environment for creating & analyzing state-machine based requirements models, with particular emphasis on mission-critical and safety-critical systems.

■ Features

- Emphasis on construction of software requirements models that can be easily read, reviewed, simulated visually, traced and analyzed
- Direct support for capturing “intent specifications” to document rationale—not only *what* and *how* but also *why*.
- Tool support for completeness in hazard analysis



SpecTRM (continued)

■ Benefits

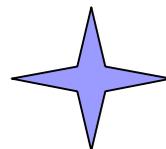
- Find consistency/completeness errors at the requirements level, where resolving errors is least costly and most effective
- Easily-learned notation enables use by domain experts
- Assist in satisfying new NASA safety standard.

■ Successes

- Derived from Prof. Nancy Leveson's work on TCAS II with NASA and FAA support
- Adopted and used by Japan Manned Space Systems Corporation
- SpecTRM-based services provided to automotive, aerospace and medical devices industry by Safeware

■ Contexts for best use

- Software-intensive, mission-critical and safety-critical systems.
- Software with complex decision-making algorithms, such as mode and state transition logic, benefit more than systems where complexity is in numerical calculations.





2. MATLAB Model Testing Tools

- MATT
 - Automated black-box test case generation, execution, and management for Matlab/Simulink models and generated code
- Reactis
 - Automated white-box test case generation and validation of Simulink and Stateflow models
- ANCT
 - Automated black-box test case generation, execution, reliability & stability analysis, and validation of Matlab/Simulink models of controllers



MATT

PoC: Joel Henry, University of Montana joel.henry@mso.umt.edu
406-243-2218 . Funding source: NASA OSMA/SARP

■ What is it: Tools for –

- Automated black-box test case generation, execution, and management for Matlab/Simulink models and generated code

■ Features

- select the entire Simulink model, or any subsystem for testing;
- generate the input values for each inport for each time step using built-in functions, user specified functions, and graphical tools;
- specify defect criteria for each outport or combination of outports;
- execute tests using simulation or test the automatically generated code;
- detect defects based on outport defect criteria; and
- analyze multiple test runs for coverage, model reliability



MATT (continued)

■ Benefits

- Quickly configure and execute tests on entire Matlab/Simulink models or any subsystem.
- Re-use tests when the models change and source code is regenerated.
- Detect and evaluate defects quickly.

■ Successes

- Used by SAIC on the STEREO (Solar-Terrestrial Relations Observatory) project (GSFC).

■ Context for best use

- Development environment where model developers use MATT consistently.
- NASA IVV Facility evaluation report specifically recommends it for unit testing.



Reactis



PoC: Steve Sims, CTO, Reactive Systems
sims@reactive-systems.com 703-534-6458

■ What is it

- Tools to automate white-box test case generation and validation of Simulink and Stateflow models.

■ Features

- User can specify structural-coverage criteria (such as branch and MCDC, as required by FAA DO-178B) in selecting test data
- Generates test data automatically from models to achieve specified coverage.
- Instruments models, collects model coverage metrics

■ Benefits

- Construct better models more quickly through requirements checking and debugging, and better code through coverage testing.

■ Successes

- Reported 25 – 75% reduction in testing costs in automotive industry.





ANCT

PoC: Fola Soares, Contek Research, Inc. at NASA DFRC

folasoares@dfrc.nasa.gov 667-276-5536

- What is it
 - Tools to automate black-box test case generation, reliability & stability analysis, and validation of Matlab/Simulink models of controllers.
- Features
 - Monte Carlo model simulation and analysis capability
 - Evaluates the time-series outputs using user-specified output evaluation functions
 - Manage test results in MySQL DB and MAT-file format.
 - Support for finding inputs and parameter values that optimize user-specified evaluation function (genetic algorithm techniques)
- Benefits
 - Simplifies analysis of control systems over entire operating envelope and under specified fault conditions.
- Successes
 - Intelligent Flight Control Systems F-15 and C-17 Projects adaptive neural flight control systems at NASA DFRC





Comparison of 3 Matlab/Simulink tools

MATT

- Black box
- User-specified inputs – functions, graphical input, data files, etc.
- MATT simulates the model (or runs the generated code)
- Reliability analysis tool analyzes multiple test output files – failure prob., MTTF, domain coverage

Reactis

- White box
- Automated generation of test cases to meet coverage conditions including MCDC
- Collects coverage metrics

ANCT

- Black box
- User-specified inputs – functions (m-files), built-ins, etc.
- Supports Monte Carlo analysis
- Finds input & parameter values that optimize evaluation function





3. Software Architecture Evaluation

- Software Architecture / Code Consistency Evaluation
 - Tools and Methodology for assuring that the code implements the intended design

- Usability and Architecture
 - Methodology for assuring that an architecture supports usability

Software Architecture / Code Consistency



Evaluation

PoC: Mikael Lindvall

mikli@fc-md.umd.edu 301-403-8972

Fraunhofer Center – Maryland

- What is it
 - Tools and Methodology for assuring that the code implements the intended design
- Features
 - Tailorable to project needs, architectural styles, design patterns, general guidelines and design rationale.
- Benefits
 - Quickly check that source code conforms to planned architecture.
 - Identify architectural violations & prevent architecture from degenerating during maintenance.
 - Assure that architecture remains flexible despite software evolution.
 - Make reviews more efficient by ensuring the architecture is accurate.



Software Architecture Evaluation (continued)

■ Successes

- Applied to several research projects and one commercial product.

■ Contexts for best use

- Best for systems designed with maintainability, reuse, flexibility or evolvability in mind.
- Can be applied to Java and C/C++ projects.
- Object-oriented systems based on design patterns, architectural styles, etc. benefit the most.
- Applications planned to have a long life benefit the most.





Usability & Architecture

PoC: Bonnie John, CMU, bej@cs.cmu.edu (412) 268-7182

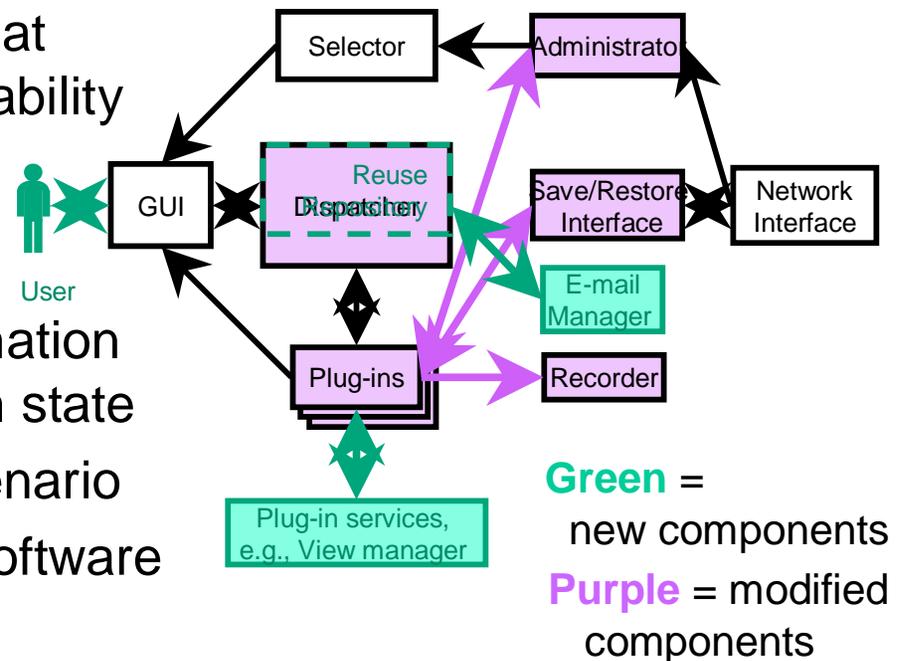
Funding: NASA Code R, Engineering for Complex Systems and Communications, Information, and Computing Technology programs, High Dependability Computing

■ What is it?

- Methodology for assuring that an architecture supports usability

■ Features

- 27 usability scenarios:
 - e.g., cancellation, information reuse, observing system state
 - Benefits of including scenario
 - Responsibilities of the software to support the scenarios
- Methods for evaluating applicability of the scenarios
- Architecture patterns that support the scenarios





Usability & Architecture (continued)

■ Benefits

- Reduce risk that the *software architecture* of an *interactive system* has to be changed due to usability concerns, or that
- architecture decisions impair usability
 - “Yikes! You mean we CAN'T CANCEL COMMANDS??!!”

■ Contexts for best use

- Early in design process
- Software with human-in-the-loop

■ Successes

- Modification of MERBoard's architecture, based on a usability analysis.



Oh no!





4. Source Code Analysis & Error Detection

- Klocwork InSpect

- Automated source code analysis tool for detecting logical code problems, metric violations and architectural violations in C, C++ and Java.

- CodeSurfer

- Intelligent source code browser/analyzer for C/C++

- Fluid

- Tool for analyzing Java source code to detect potential race conditions and, in some cases, assure their absence



Klocwork InSpect

PoC: Don Hewitt, Klocwork, Inc.

don.hewitt@klocwork.com 925-461-5347

- What is it?
 - Automated source code analysis tool for detecting logical code problems, metric violations and architectural violations in **C** and **C++**
- Features
 - Detects 50 pre-defined defect classes
 - Customizable to include user-defined defect classes
 - Can be integrated into organization's build & issue-reporting process
 - Relatively high accuracy (identifies real defects with relatively few false alarms) compared to most other source code analysis tools
- Benefits
 - Identification of defects early in the development lifecycle → reduced cost to fix
 - High accuracy → less time spent on filtering false alarms



Klocwork InSpect (continued)

■ Successes

- Positive evaluation by Gerard Holzmann at JPL on several example applications including 400 K non-comment source code flight application.
- HP and other large software developers

■ Contexts for best use

- Integrated into the development environment
- Target applications should follow good programming practices guidelines



CodeSurfer



PoC: Mark Zarins, GrammaTech, Inc.

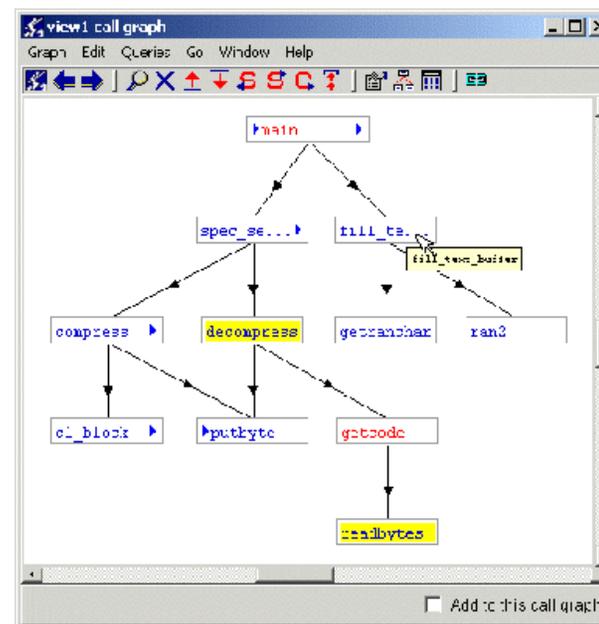
mzarins@grammatech.com 408-688-1243

■ What is it: Intelligent **C/C++** source code browser for—

- Code Inspection
- Debugging
- Safety/Security auditing
- Documentation
- Reverse engineering

■ Features

- Interactive, graphical reports
 - Trace data flow backward and forward through code
 - Display what variables a pointer can point to
 - Highlight code that affects selected statement(s) and/or variable(s)
 - Change impact analysis, etc.
- Based on program slicing and pointer analysis technologies
- Commercially supported product





CodeSurfer (continued)

- Benefits
 - CodeSurfer makes it easy to analyze and understand code.
- Successes
 - NASA, Mitre, MIT, Thales, Network Associates
 - Successful 2004 Research Infusion collaboration at JSC:

“Our group analyzes many mission-critical software projects to reduce defects and ensure that the software meets requirements.

We conducted a formal study... to see if CodeSurfer could improve our software inspections. In parallel to our normal inspection process, an independent team used CodeSurfer.

We found that CodeSurfer reduced inspection time by an average of 40%.

In addition, when using CodeSurfer the number of defects found increased by an average 116%.”



CodeSurfer (continued)

■ Contexts for best use

- Code inspection/reviews, and debugging
- Need compilable C source code; build application with CodeSurfer using one of the standard C/C++ compilers provided.
- Best applied on applications of up to 100K – 500K LOC.





Fluid

PoC: William Scherlis, CMU

scherlis@cs.cmu.edu (412) 268-8741

Funded by NASA High Dependability Computing Program

- What is it
 - Tool for analyzing **Java** source code to detect potential race conditions and, in most cases, assure their absence
- Features
 - Can provide positive assurance of absence of race conditions
 - Uses Javadoc-style declarations of design intent as program annotations
 - Uses static analysis to assess consistency of the code and the models expressed using the program annotations
 - Integrated into Eclipse open source development environment
- Benefits
 - Provides static assurances for critical multi-threading properties that are difficult to assess using traditional testing & inspection.
 - Reduces likelihood of introducing race conditions.



Fluid (continued)

■ Successes

- Experimentally applied to wide variety of Java production systems and components, including commercial applications, JPL applications, Sun's Electric, and NASA's CTAS
- Found faults that can trigger race conditions (not false alarms) in nearly all larger systems, including widely used library code.

■ Context for best use

- Most effective on Java systems that are organized into subsystems (for example, a 350KLOC system decomposable into subsystems sized 50KLOC or less).
- Focus is on both lock-based concurrency and non-lock concurrency (e.g., as used in GUIs and no-heap real-time threads).
- For new or old development, users must be willing to provide the required program annotations





5. Linux-based RTOS

- RTLinuxPro
 - POSIX hard-real-time, Linux-based operating system



RTLinuxPro

PoC: Michael Cravens, FSMLabs, Inc.
mcravens@fsmllabs.com 972-693-7799

- What is it
 - POSIX hard-real-time, Linux-based operating system
- Features
 - POSIX 1003.13 threads API
 - Large number of built-in methods for real-time control applications including safe methods for non-real-time processes under Linux
 - Memory-protected real time threads and frame scheduler
 - Library emulating VxWorks functions
 - Decoupled architecture for RT and non-RT software
- Benefits
 - Decoupled architecture prevents non-RT software from interfering with the operation of the RT system
 - Low-microsecond worst case real-time plus standard Linux development and runtime environment
 - Potentially reduced cost compared with VxWorks



RTLinuxPro (continued)

■ Benefits (continued)

- Leverages Open Source resources in a hard-real-time environment

■ Successes

- Development and test of P&W JSF F-134 Jet Engine used a machine-in-the-loop simulator hosted on RTLinux
- Used at Sandia Laboratories
- ATOS-Origin has ported software for the ESA.
- Experimental uses and evaluation of Linux-compatible real time operating systems conducted at GRC funded by OSMA
- Other industrial customers: BBN, Lucent, Harris

■ Context for best use

- Where there is a need for the combination of a full UNIX system and hard-real-time control.





6. Software Reliability Estimation

- SMERFS³
 - Software reliability analysis tool

- CASRE
 - Software reliability analysis tool



SMERFS³: Statistical Modeling and Estimation of Reliability

Functions for Systems

PoC: William Farr, Naval Surface Warfare Center Dahlgren Division

William.Farr@navy.mil 540-653-8388

■ What is it

- Tool for determining a variety of reliability and availability measures at either the component level (hardware or software) or the system level of any software-intensive system.

■ Features

- Uses observed failure data to fit reliability models to the data and then use the models to estimate and predict reliability measures.
- Models include 11 of the most-used in the literature
- Measures include: remaining number of faults, mean time to failure, operational reliability for a specified time, testing time required to achieve desired level of quality, and system availability.
- Supports trade and sensitivity studies.



SMERFS³ (continued)

■ Benefits: Helps determine –

- testing resource allocation
- when testing should stop
- when the software component should be re-engineered
- estimate of availability for systems.
- Trade and sensitivity studies allow improved judgment of system quality.

■ Successes

- Earlier version of tool used by Lockheed Martin as reliability check on flight control software before each Shuttle flight.
- Analyzed reliability & availability of ground-based software system and a satellite system at GSFC
- Assessed quality of Navy's TRIDENT Fire Control, AEGIS, and components of TOMAHAWK software.



SMERFS³ (continued)

■ Contexts for best use

- Software-intensive systems where reliability is a critical factor
- Need data on faults such as fault severity, testing intensity (e.g., number of test cases run), time/date of failure occurrence, time to fix the fault (for estimating system availability).
- Some level of statistical sophistication is required to interpret SMERFS³ results.





CASRE: Computer-Aided Software Reliability

Estimation *Allen Nikora, JPL* Allen.P.Nikora@jpl.nasa.gov 818.393.1104

■ What is it

- Tool for estimating and forecasting software reliability based on the failure history of a software system during test.
- Similar capabilities as SMERFS³

■ Successes

- USAF Operational Test & Evaluation Center – funded development and use on software systems developed for the Air Force.
- Sun Microsystems, HP Printer Division, Raytheon, Motorola
- > 400 copies downloaded from Open Channel Foundation

■ Comparison with SMERFS³

- Same mathematical libraries as in SMERFS
- Implements trend tests that help determine whether it is appropriate to apply software reliability models
- Can combine model results according to user-specified static or dynamic weighting schemes to improve model accuracy.





7. Software Process Improvement

- Orthogonal Defect Classification
 - Software defect classification system with defect data feedback for process improvement

- Formal Inspection Training Course
 - Cost-effective methodology, discussed earlier, for early detection of software defects



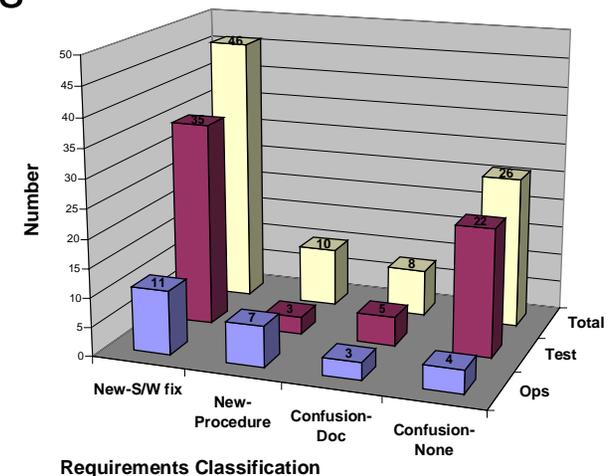
Orthogonal Defect Classification

PoC: Robyn Lutz, JPL

Funding: Office of Safety and Mission Assurance, Software Assurance Research Program; and National Science Foundation

■ What is it

- Software defect classification system with defect data feedback for process improvement
- First developed ~1990 by IBM, now widely used in industry
- Provides generalized format for defect logs
- It “looks at the forest, not the trees” to identify defect patterns of concern.





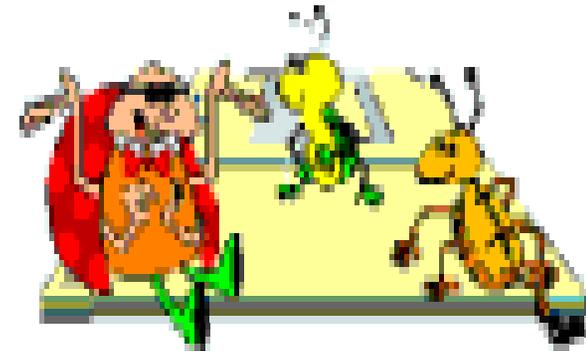
Orthogonal Defect Classification (continued)

■ Features

- Language & platform independent
- Produces customizable Excel graphs
- Much NASA expertise

■ Benefits

- Provides quantitative basis for process improvement
 - Establishes a baseline for patterns of software defects
 - Much less expensive than root-cause analysis
- Provides guidance in allocating funds for post-launch maintenance
- Enables effective corporate memory
- Useful to single project or to organization



Orthogonal Defect Classification



Orthogonal Defect Classification (continued)

■ Successes

- Analyzed ~800 testing problem reports from Mars Exploration Rover
 - Identified mechanisms by which requirements changes occur and are resolved during testing and operations.
- 2004 Research Infusion collaboration with JPL ground software project
- Adopted by companies such as IBM, Motorola, Telcordia, Cisco, and Nortel

■ Contexts for best use

- Teams that want to incorporate improved defect metrics into their development or maintenance process.





Next Step

- If you're interested in a collaboration involving a Research Infusion technology, check out the collaboration proposal process at

<http://ic.arc.nasa.gov/researchinfusion/>

- We will help broker matches of technology and software developers.



Software Engineering Research Infusion Technologies for 2006 Collaborations



<http://ic.arc.nasa.gov/researchinfusion>

- Requirements capture, modeling and testing: SpecTRM
- Matlab Testing Tools MATT, Reactis, ANCT
- Software architecture evaluation: Architecture/Code Consistency Evaluation, Usability & Architecture
- Source code analysis and error detection: Klocwork InSpect, CodeSurfer, Fluid
- Real time operating system: RTLinux Pro
- Software Reliability Estimation: SMERFS³, CASRE
- Software Process Improvement: Orthogonal Defect Classification

