National Aeronautics and Space Administration

Dependability Quantification and Assurance of Mission-critical Software Systems

NASA OSMA Software Assurance Symposium
August 4-12, 2011

PI: Allen Nikora: Jet Propulsion Laboratory, California Institute of Technology
Co-Is: Prof. Kishor Trivedi: Duke University
Prof. Gianfranco Ciardo: UC Riverside

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program. This activity is managed locally at JPL through the Assurance and Technology Program Office.

Agenda

National Aeronautics and Space Administration

- Problem/Approach
- Relevance to NASA
- Current Capability and Results
- Planned Capability
- Technical Solutions and Remaining Technical Challenges
- Backup Information

Problem/Approach (1)

- **Software systems continue to increase in size, complexity, and importance to the success of NASA missions. For the foreseeable future, software systems will continue to harbor residual faults.**
- **Problem is to minimize number of introduced faults and number of residual faults that manifest as significant failures.**
  - Identify potential for fault introduction during early stages of development.
  - Design and implement mechanisms for tolerating software-related defects observed during previous and currently operating missions.
  - Develop mechanisms for preventing/mitigating effects of residual faults that manifest as failures during mission operations.
    - Run-time model checking
    - Rejuvenation (controlled full or partial reset to revert to known safe state).

Problem/Approach (2)

- **Goal: develop techniques to handle the three classes of faults:**
  - Bohrbugs: consistently manifest themselves under well-defined conditions.
  - Mandelbugs: cause hard to reproduce failures.
  - Aging-related bugs: manifestation rate increases with execution time (e.g., memory leaks, improper termination of processes).
- **Plan to investigate model-checking and stochastic analytic modeling to demonstrate in a coordinated manner their applicability to and effectiveness for mission-critical software applications.**
  - Advanced model-checking to find Bohrbugs and concurrency-related Mandelbugs prior to runtime
  - On-line model-checking as a last line of defense against residual bugs.
  - Proactive failure avoidance techniques (e.g., software rejuvenation) for aging-related bugs to increase TTF
  - Reactive recovery: main mitigation for failures caused by Mandelbugs
    - Investigate analytical model-based fast recovery techniques to reduce TTR.

Problem/Approach (3)

Approach

1. Phase 1

- Analyze problem reports from current and historic JPL/NASA missions to identify most frequently encountered failure mechanisms in terms of three classes of faults
- Develop abstract models for correctness properties of collaborating system critical elements (e.g., recovery schemes)
  - Apply design-time model-checking using the Saturation algorithm and multi-way decision diagrams (MDDs).
- Develop quantitative dependability models to study and reduce the time to recover from failures.

Approach (cont'd)

2. Phase 2

- Identify and develop techniques to recover from Mandelbug-induced failures.
- Apply model checking:
  - At design time to verify abstract models of critical components to show that they satisfy correctness properties.
  - On-line when multiple applications and recovery procedures need to be verified in concert at deployment time.
- Develop and pilot training material on quantitative dependability modeling; provide guidelines, best practices of model-checking and analytic modeling.

Problem/Approach (5)

Approach (cont'd)

3. Phase 3

- Identify and develop appropriate techniques to proactively recover from aging-related bugs.
- Refine dependability models to include mitigations for aging-related bugs and reevaluate the dependability model.
- Extend abstract models to include mechanisms for recovering from aging-related bugs.
- Perform bounded run-time model-checking using linear temporal logic to verify the absence of specific faulty behavior within and across applications and to proactively steer a system towards "safer" execution scenarios.

Relevance to NASA (1)

National Aeronautics and Space Administration

- **Residual software faults that manifest as failures during operations increase likelihood of adverse mission impact:**
  - Software complexity is increasing in response to increasingly complex mission goals and more complex spacecraft.
  - Software is used to control more spacecraft subsystems/components.
  - Previous work indicates:
    - Software failure intensity for spacecraft systems may be increasing.
      - A. Nikora, "Software Anomaly Trends for Space Missions", JPL internal presentation, October, 2007 (unpublished).
    - Proportions of different types of faults (Bohrbugs/Mandelbugs/Aging-related defects) have not changed significantly from mission to mission.
      - M. Grottke, A. Nikora, K. Trivedi, "An Empirical Investigation of Fault Types in Space Mission System Software," proceedings of 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010), Chicago, IL, Jun 28 – Jul 1, 2010.
- **Techniques for identifying software faults during development are likely to be more effective if they are targeted to the most frequently occurring types of faults.**
- **Recognize that residual faults will occur in operational systems, and implement mechanisms to pro-actively avoid or mitigate their effects.**

Current Capability and Results (1)

- **Analyze problem reports from current and historic JPL/NASA missions to identify most frequently encountered failure mechanisms in terms of three classes of faults.**
  - Analyzing problem reports for same set of missions being analyzed for the task "Predicting Fault Types and Fault Links for Systems using Historical Data."
    - Also analyzing problems reported during spacecraft integration testing – dataset is somewhat larger than that for anomalies observed during mission operations.
    - A large number of new problem reports for MSL are available for analysis.
    - Approximately half-way through analyzing anomaly reports for ground software.
      - Many software-related failures classified as Bohrbugs appear to be:
        - Incorrect value for data initialization.
        - Incorrect parameter format/order when invoking a function.
        - Incorrect exception handling (unanticipated fault condition, wrong response to known fault condition). This last is consistent with earlier work by Lutz and Mikulski.

Current Capability and Results (2)

- **Develop abstract models for correctness properties of collaborating system critical elements (e.g., recovery schemes)**
  - Apply design-time model-checking using the Saturation algorithm and multi-way decision diagrams (MDDs).
  - In progress.
    - EDL for recent missions.
    - Onboard sequencing management/control:
      - Rollback/rollforward.
        - Rollback to checkpoint up to “N” times if fault encountered.
        - Noncritical mode: Suspend/terminate sequence if fault encountered more than “N” times.
        - Critical mode (e.g., orbit insertion, trajectory correction): Proceed past fault if fault encountered more than “N” times.
      - Sequence start, pause, resume, terminate.
        - Sequence starts/pauses/stops/resumes after it is commanded to do so.
        - No unexpected duplicate runs of sequence.
        - Proper pause/termination/resumption of child sequences.

Current Capability and Results (3)

- **Develop quantitative dependability model.**
  - Will use results of problem report analysis. In progress.
- **Develop training materials**
  - Classroom, on-line training.
  - Modeling guidelines.

Planned Capability (1)

- **Mechanisms for preventing and mitigating effects of specific types of faults**
  - During development
    - Identify most common types of failure mechanisms associated with Bohrbugs, Mandelbugs, and aging-related bugs. (in progress)
    - Develop correctness properties, abstract models for critical software elements. (in progress)
  - During operations
    - Proactively recover from aging-related bugs (e.g., rejuvenation)
    - Use bounded run-time model checking to verify absence of specific faulty behavior.
    - Proactively steer system towards “safer” operational scenarios.

Planned Capability (2)

- **Quantitative dependability model**
  - Dependability estimation and allocation.
  - Model dependability at system, component levels.
    - Hardware and software components.
    - Human dependability?
      - Not in original task scope, but results from earlier work indicate that up to 20% of anomalies observed for ground-based support software are related to operator error or misunderstanding.
      - Possible relation to “Command Process Modeling & Risk Analysis” task.
  - Include mitigations for Mandelbugs, aging-related bugs.

Planned Capability (3)

- **Training material for engineers.**
  - Classroom instruction.
  - On-line instruction (e.g., SATERN).
  - Guidelines.

Technical Solutions and Remaining Technical Challenges (1)

National Aeronautics and Space Administration

- **Failure mechanisms**
  - Ensure representative sample of anomaly reports for analysis.
    - Preference given to “newer” missions.
    - Analyze reports for missions developed in-house as opposed to those that are contracted out.
    - Insufficient levels of detail in some problem reports prevent identification of failure mechanisms.
      - May introduce some bias into sample.
- **Correctness properties, abstract models of critical system elements.**
  - Collaboration with on-going AADL work may be possible.

Backup Information

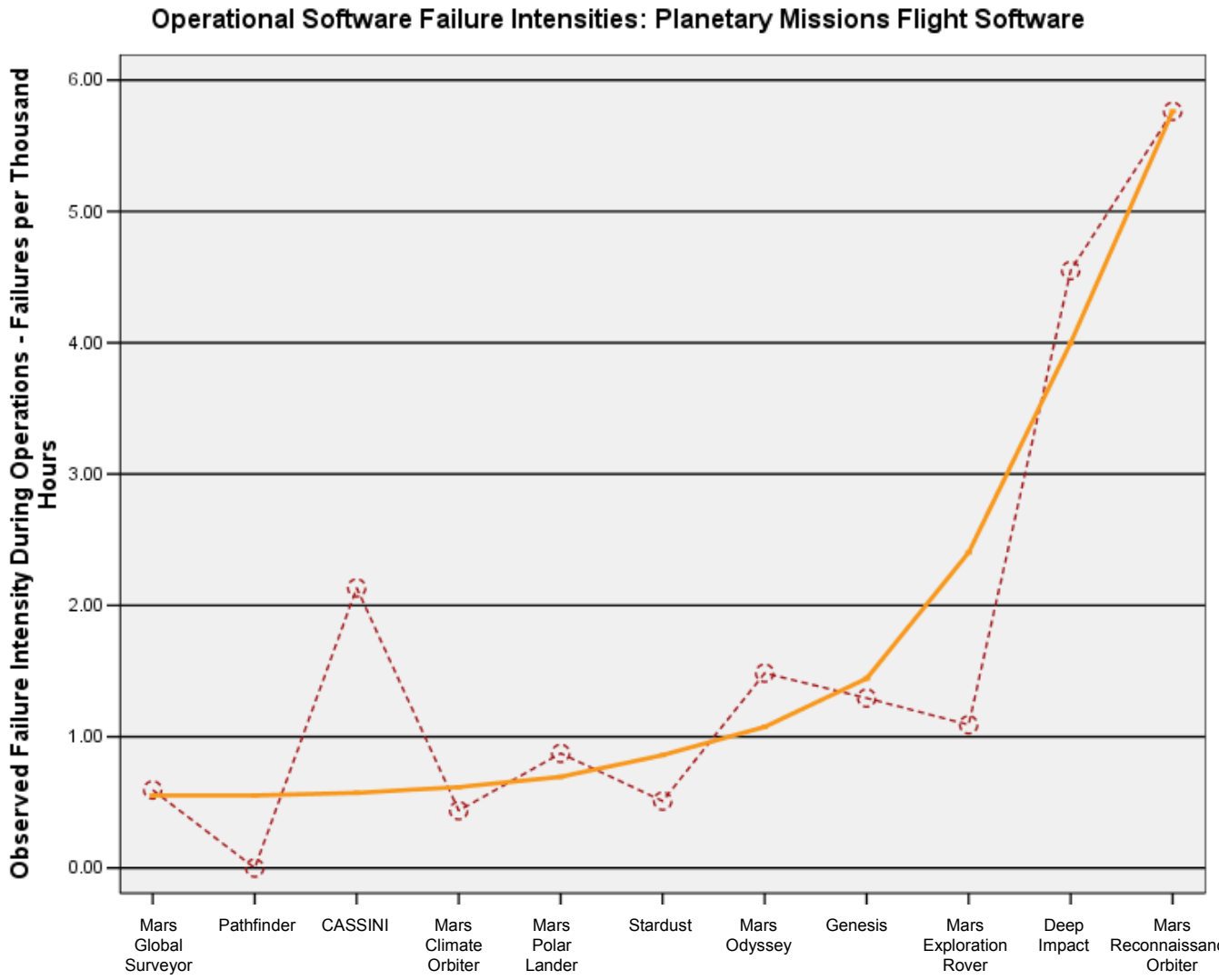
National Aeronautics and Space Administration

- Evidence for Increasing Software Failure Rates
- Trends in Fault Type Proportions for Flight Software
- Saturation Algorithm for Model Checking

Increasing SW Failure Rate?

Planetary Missions Flight Software

National Aeronautics and Space Administration



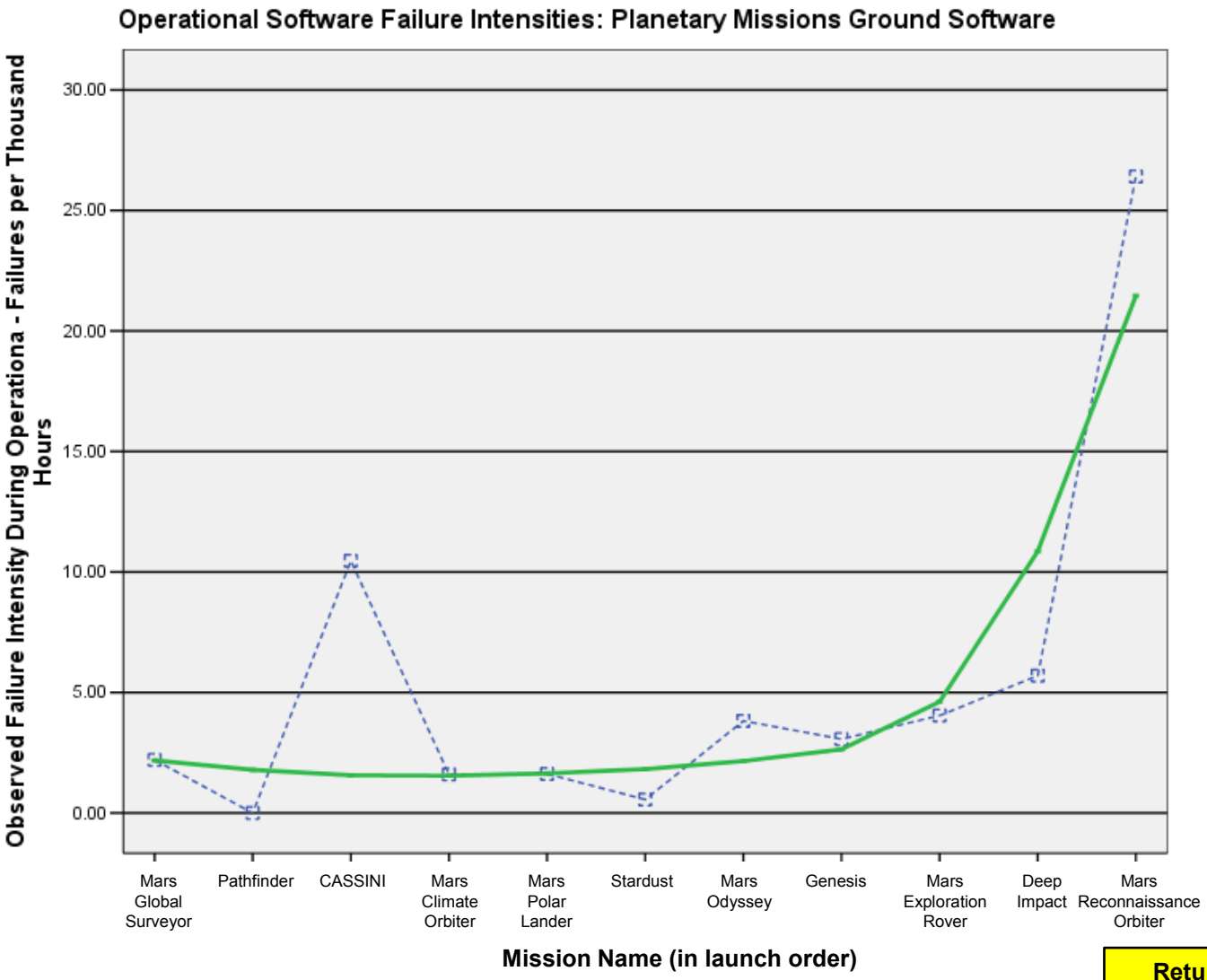
The interval between the first and last launch on this plot is 8.76 years. The interval between successive launches ranges from 23 to 790 days.

Return to Relevance to NASA

Increasing SW Failure Rate?

Planetary Missions Ground Support Software

National Aeronautics and Space Administration



The interval between the first and last launch on this plot is 8.76 years. The interval between successive launches ranges from 23 to 790 days.

Return to Relevance to NASA

Increasing SW Failure Rate?

Planetary Missions Ground Support Software

National Aeronautics and Space Administration

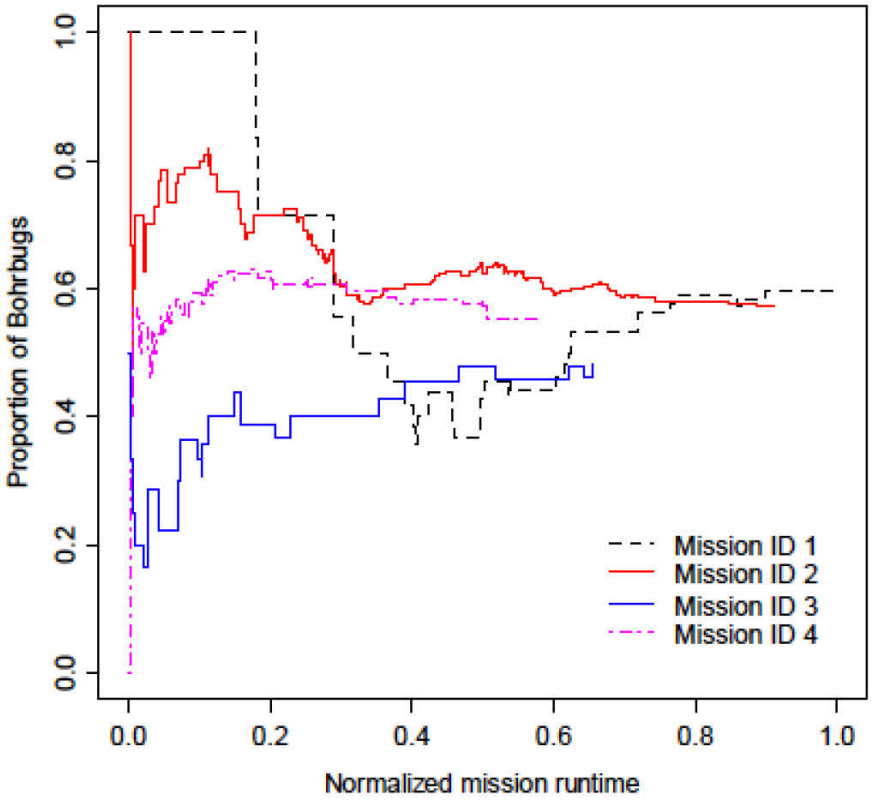
- **Launch dates for missions included in failure rate analysis:**
  - Mars Global Surveyor: 07-Nov-1996
  - Mars Pathfinder: 04-Dec-1996
  - Cassini-Huygens: 15-Oct-1997
  - Mars Climate Orbiter: 11-Dec-1998
  - Mars Polar Lander: 03-Jan-1999
  - Stardust: 07-Feb-1999
  - Mars Odyssey: 07-Apr-2001
  - Genesis: 08-Aug-2001
  - Mars Exploration Rovers (Spirit): 10-Jun-2003
  - Mars Exploration Rovers (Opportunity): 07-Jul-2003
  - Deep Impact: 12-Jan-2005
  - Mars Reconnaissance Orbiter: 12-Aug-2005

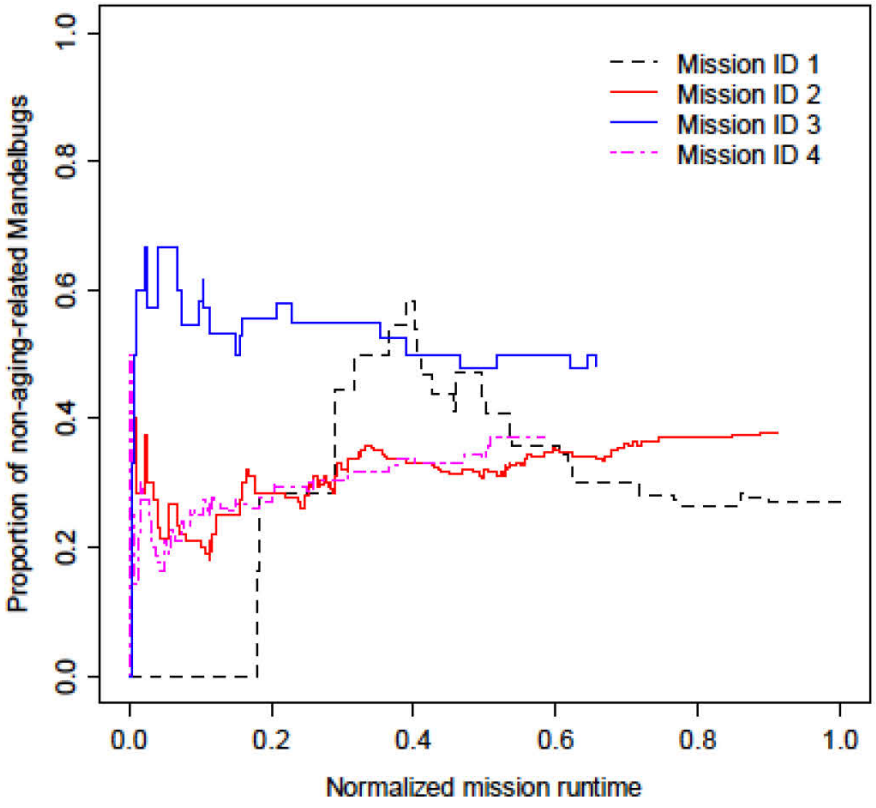
Return to Relevance to NASA

Trends in SW Fault Type Proportions

Planetary Missions Flight Software

National Aeronautics and Space Administration





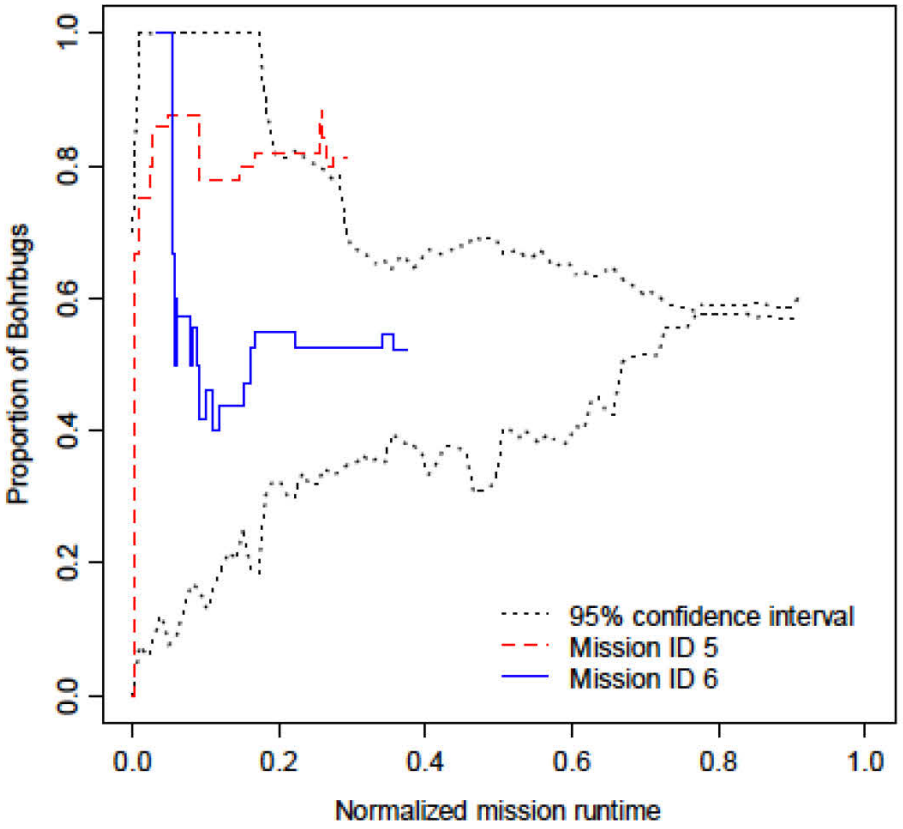
- Fault Type Proportions vs. Runtime for Four Earlier Missions (of 8 missions analyzed)
- Result: The proportion of Bohrbugs seems to settle at around the same value. Such a convergence to similar values is less obvious for the other fault types.

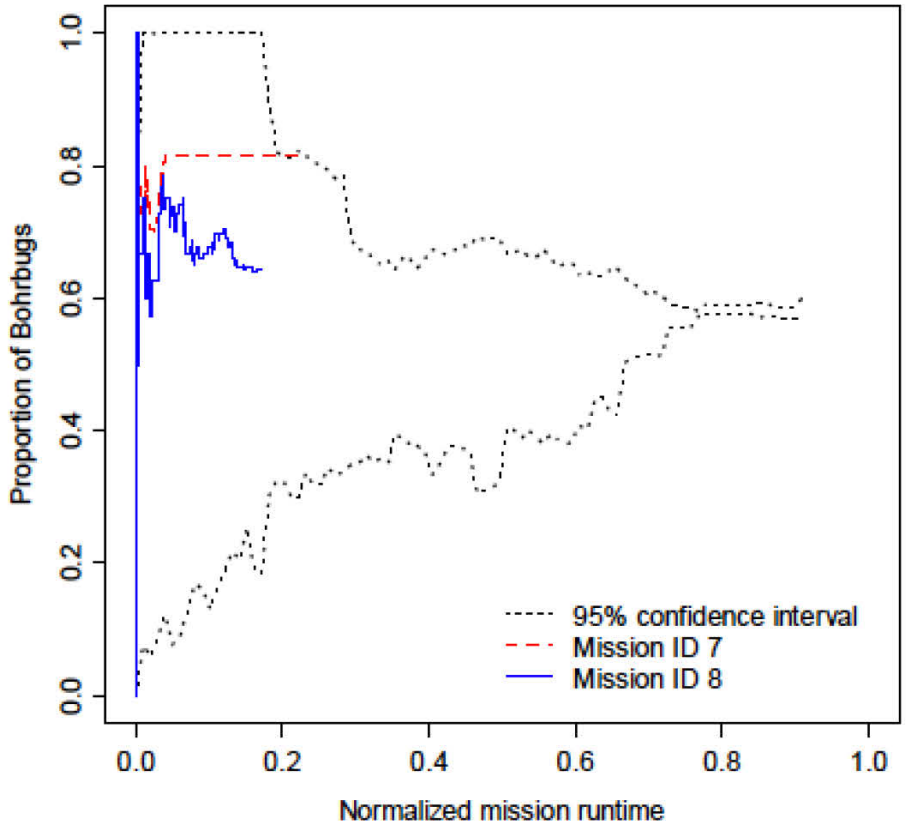
Return to Relevance to NASA

Trends in SW Fault Type Proportions

Planetary Missions Flight Software

National Aeronautics and Space Administration





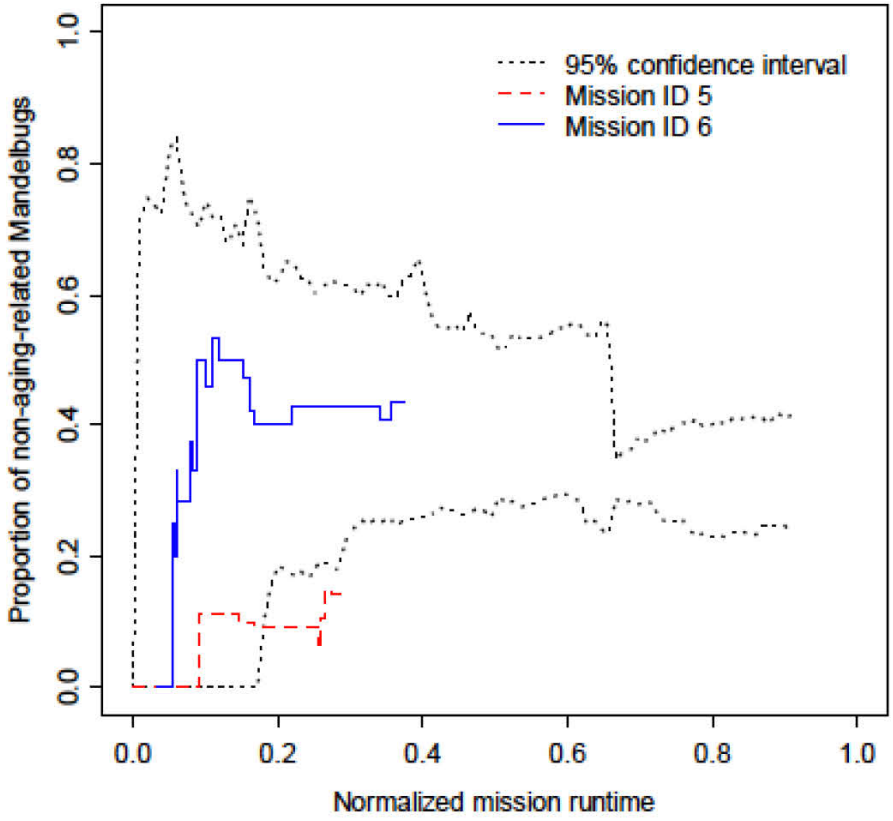
- Confidence intervals obtained for the earlier four missions, and Bohrbug proportions for the four later missions.
- Result: For two of the four later missions, the Bohrbug proportions observed are higher than expected.

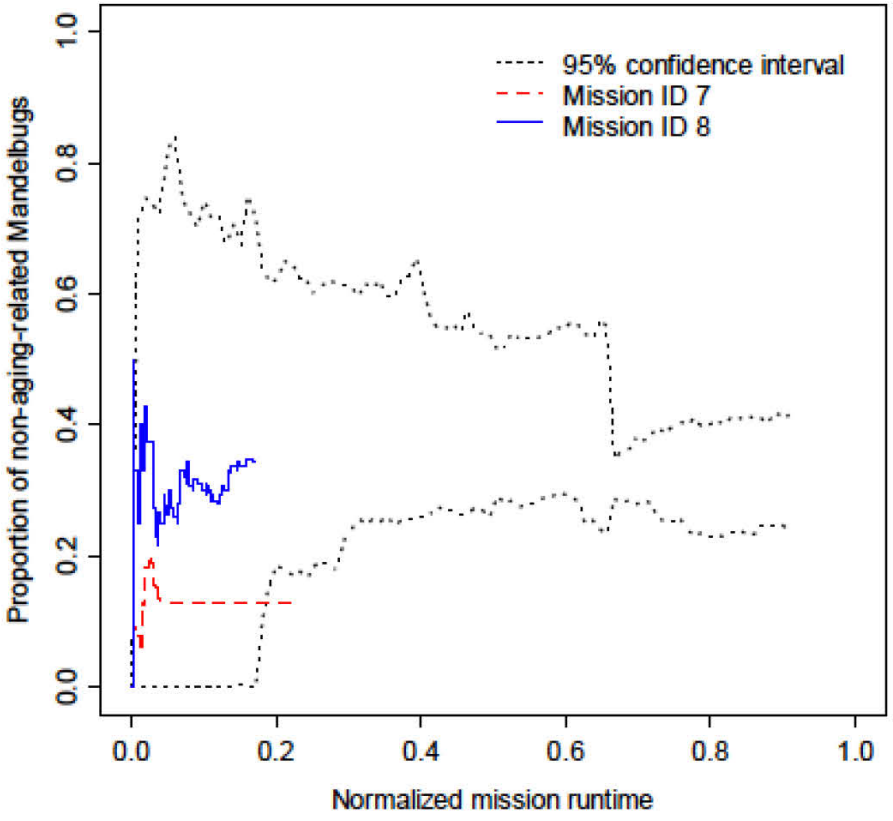
Return to Relevance to NASA

Trends in SW Fault Type Proportions

Planetary Missions Flight Software

National Aeronautics and Space Administration





- Confidence intervals, and proportions of non-aging-related Mandelbugs for the four later missions.
- Result: The proportions of non-aging-related Mandelbugs are lower than expected for two of the four later missions.

Return to Relevance to NASA

Saturation Algorithm for Model Checking

National Aeronautics and Space Administration

- **Efficient technique to generate state spaces of asynchronous systems for model checking[1].**
  - Applicable to complex critical systems (e.g., spacecraft on-board control software) for which state space volume can be very large.
  - Based on Multi-valued Decision Diagrams (MDDs).
    - For event-based asynchronous systems, each event updates just a few components of a system's state vector.
    - Requires only the application of local next-state functions and the local manipulation of MDDs.
      - Compare to classic BDD-based techniques which construct state spaces by iteratively applying a single, global next-state function which is itself encoded as a BDD.
    - Experimental results show significant improvements in speed and memory over other state-space generators.

1. G. Ciardo, G. Luettgen, and R. Siminiceanu. Saturation: an efficient iteration strategy for symbolic state-space generation. In Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001), pages 328-342. Genova, Italy, Apr. 2001. Springer-Verlag.

Return to Current Capability (2)