Software Reliability: Requirements Issues in Large Software Intensive Systems

Presented to
2007 Systems and Software Technology Conference
Tampa, FL

Myron Hecht,
The Aerospace Corporation
El Segundo, CA

June, 2007

THE AEROSPACE CORPORATION

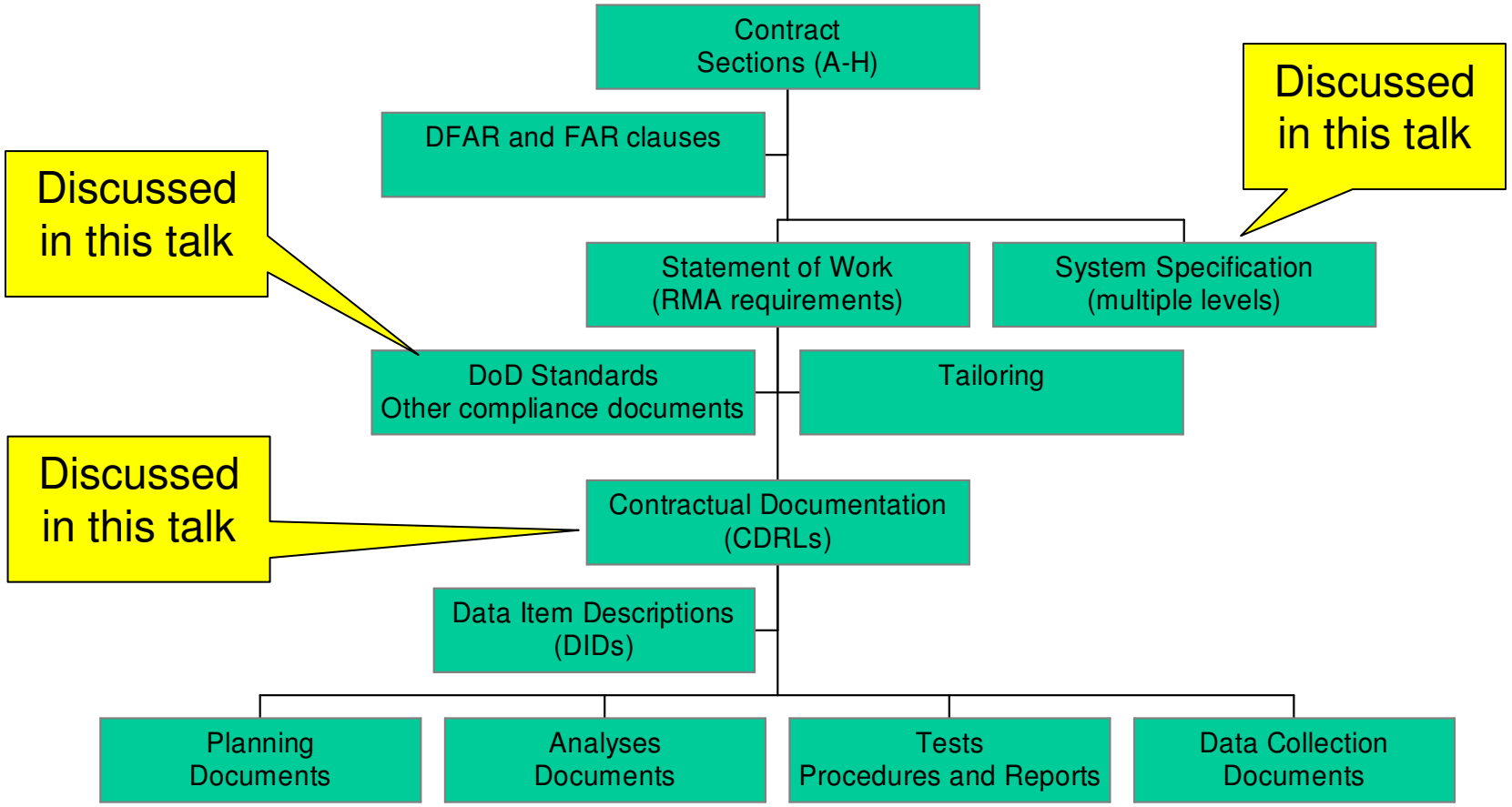
Outline

- Problem Domain
- Motivation
- Failure Taxonomy
- Reliability Metrics
- Selected Problems and Solutions in Statements of System Requirements (Specifications)
- Selected Problems and Solutions in Contractual Requirements (Deliverables)

Problem Domain

- **Software-intensive systems for DoD, NASA, and NOAA programs**
- **Acquired using FARs, DFARs, and associated contractual instruments**
- **DoD or NASA standards for RMA/Safety processes**
- **Consequences of failures have greater severity than organizational inconvenience or rare interruption of service**
  - Excludes most “IT” applications
  - Nevertheless, many such systems utilize the same hardware, system software, middleware, and applications of traditional “IT” applications

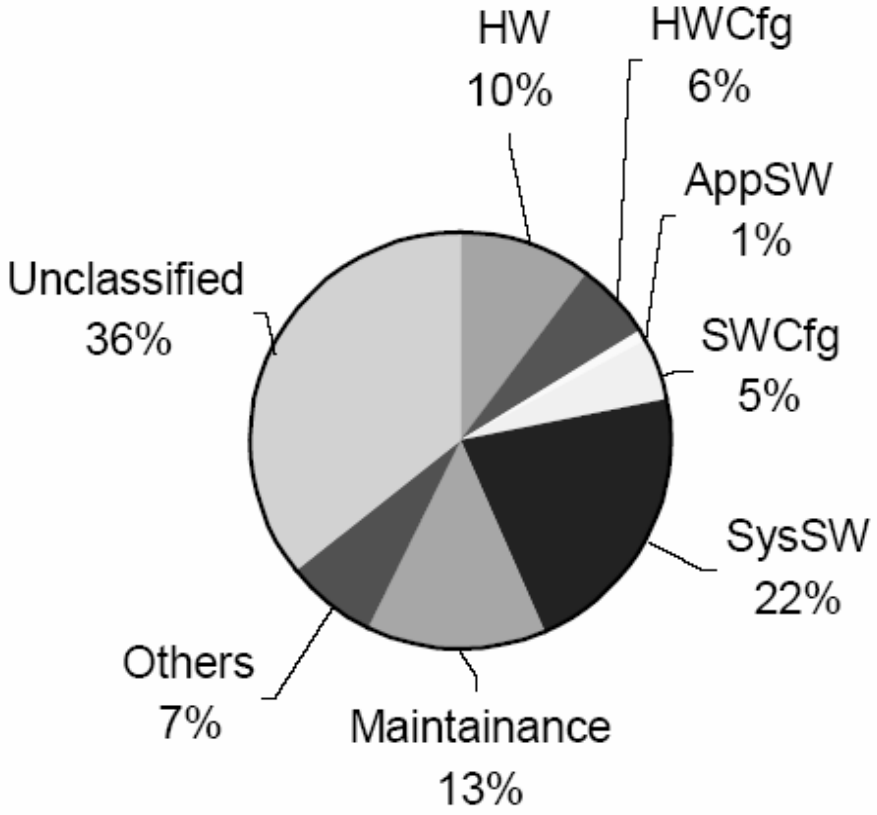
Usual U.S. Federal Government Contract Structure



Importance of Software Failures

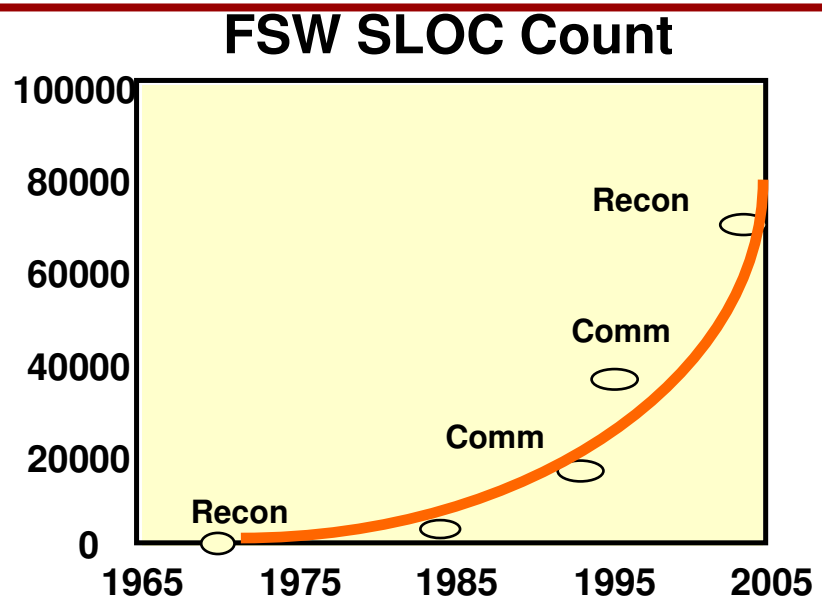
503 servers running in a production environment over a four-month period. The event logs contain only system reboot information.

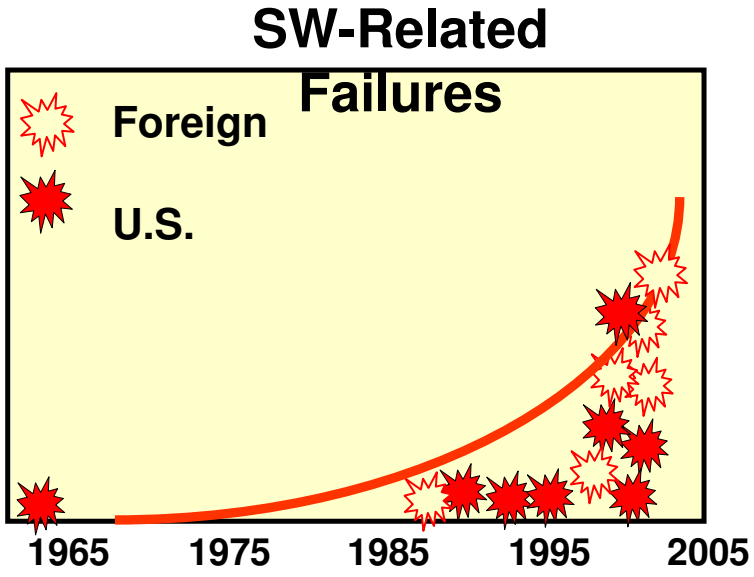
Hardware failures occurred less frequently than indicated by downtime because of relatively long recovery time from hardware failures



Source: J. Xu, Z. Kalbarczyk, and R. K. Iyer. Networked Windows NT System Field Failure Data Analysis. Proc. 1999 Pacific Rim Int'l Symp. Dependable Computing, IEEE CS Press, Los Alamitos, Calif., 1999., available from http://citeseer.ist.psu.edu/xu99networked.html

Importance of Software Failures: Space Systems





Over half of failures between 1998 and 2000 involved software

FSW SLOC = Flight Software Source Lines of Codes

- **Chart presented by Paul Cheng, *Ground Software Errors Can Cause Satellites to Fail too*, GSAW 2003**

A Requirements-Oriented Taxonomy of Software Failures

- **Manifestation**
  - Deterministic
  - Random
- **Severity**
  - Recoverable
  - Degradation
  - Non-recoverable

Manifestation

- Deterministic ("Bohrbugs")
  - Analogous to workmanship defects in hardware
  - Repeatable with known or determinable root cause under control of developer or user
    - Incorrect/ambiguous requirement,
    - Configuration Management,
    - Coding error,
    - SW Installation,
    - SW Maintenance
- Random ("Heisenbugs)
  - Not repeatable
    - crash, hang, late response
    - many failures fixed by reset
  - Due to transient system states (timing, buffer overflows, long queues causing missed deadlines, memory leaks)

Requirements Impact

- **Bohrbugs**
  - Can be prevented through disciplined development process (analogous to quality initiatives in hardware)
  - Address through system engineering, software development process, and test program requirements
- **Heisenbugs**
  - Can not be eliminated through traditional quality processes
  - Address through
    - Allocation of quantitative requirements from system to software levels
    - Integrated hardware/software modeling to assess conformance with quantitative requirements
    - Reliability program that includes software as well as hardware
    - Failure data collection program that collects *all* failures (not only reproducible ones), operating time, and recovery times

Severity: Explanation by Analogy (space vehicles)

- **Non-recoverable Failures**
  - Loss of power distribution
  - Maneuvering jet valve fails to close
  - Antenna deployment failure
  - Depletion of Expendables (e.g., Propellant)
- **Degradation Failures**
  - Wearout
    - Gimbals, Thermostats, Solar Cells
- **Recoverable Failures**
  - CPU lockup due to SEU
  - Low power

Non-recoverable Software Failure Examples

- Mars "Spirit" rover to execute any task that requested memory from the flight computer[1],
- Unanticipated descent of the Mars Climate Orbiter spacecraft into the Martian atmosphere due to a navigation system unit conversion defect[2],
- Crash of the Mars Polar Lander onto the Martian surface due to a premature shutdown of its descent engines.[3]
- Failure of first launch of the Ariane 5 booster; cause was traced to a variable overflow that affected software running in both channels of its dual redundant inertial reference system (IRS).[4]

[1] Glenn Reeves, Tracy Neilson & Todd Litwin, "Mars Exploration Rover Spirit Vehicle Anomaly Report," Jet Propulsion Laboratory Document No. D-22919, May 12, 2004.

[2] Mars Climate Orbiter Mishap Investigation Board, *Phase I Report*, November 10, 1999, available from ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf, last visited January 23, 2005

[3] *Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions JPL Special Review Board*, March, 2000 available from http://klabs.org/richcontent/Reports/NASA_Reports/mpl_report_1.pdf, last visited January 23, 2005

[4] ARIANE 5 Flight 501 Failure Report by the Inquiry Board, "Lions Report", Paris, 19 July 1996 available from http://www.dcs.gla.ac.uk/~johnson/teaching/safety/reports/ariane5.html, last visited January 23, 2005

Degradation Software Failure Example

- **European Space Agency's Huygens Probe lost half of its image data due to a missing line of code.[5]**

[5] J. Watson, "Veteran software makes it to Titan: Two tiny software errors nearly a decade ago almost lost the historic pictures of Saturn's moon" *Computing*, January 26th, 2005, available on line at http://www.computing.co.uk/analysis/1160783

Recoverable Software Failure Examples

- **Space Vehicle (SV)**
  - Incorrect SV Ephemeris data download (due to modifications that had been made 7 years earlier)
  - Spurious SV failure indication and switchover (due to incorrect database values and timing window when hardware telemetry was vulnerable to misreads)
- **Ground System**
  - Failure to switchover after failure in (downed) primary channel due to incorrect configuration data values
  - Failure to re-initialize after a failure due to incorrect procedures (numerous instances)

Requirements Impact

- **Recoverable failures**
  - If software is mature, should be primarily “Heisenbugs”
  - Address through quantitative reliability/availability requirements
  - Restorations times and recovery probability requirements (probably allocated by the development contractor) are particularly important
  - Failure detection and diagnostics requirements should be derived from the FMEA process
- **Non-recoverable failures**
  - Addressed through system safety process
    - System safety process must include both hardware and software
    - Must be coordinated with software development process
      - Many similarities between system safety and software development processes
  - Necessary condition: system level requirements must be complete
    - Requirements need to specify what system “*shall not*” do
    - Major root cause of software failures causing death or major destruction: missing requirements (Leveson)
  - Treat like “Bohrbugs on steroids”

Requirements Impact (continued)

- **Degradation failures**
  - Somewhere in between recoverable and non-recoverable failures
    - More severe degradations treated as non-recoverable failures
    - Less severe treated as recoverable failures
  - Also addressed through software upgrade requirements
    - Transition/fallback requirements
    - Off-line test and simulation capabilities
    - Transition Support facilities

Metrics

- **Reliability**
  - Mean Time Between Failures and Mean Uptime
  - Mean Time to Restore and Mean Downtime
  - Availability
  - Design Life
- **Software development/quality**
  - Requirements traceability (childless requirements, orphan requirements) and stability (percent changed)
  - Verification completeness (reviews, inspections, tests)
  - Distribution of open and closed problem reports (severity, component, time open, etc.)
  - Many others

Mean Time Between Failures (MTBF) and Mean Uptime (MUT)

- **Definition:**
  - MTBF: Operating Time/Number of Failures
  - MUT: Operating time/failures causing outages
- **Associated metric (for redundant systems)**
  - Proportion of common mode failures (i.e., failures that defeat the redundancy provisions)
- **Related primarily to Heisenbugs**
- **What requirements must clarify**
  - Definition of Operating Time
  - Definition of Failure
  - Data collection and analysis process
    - Example: Treatment of scheduled outages
  - Analytical Procedures

Mean Time To Restore (MTTR) and Mean Downtime (MDT)

- **Definition:**
  - MTTR: Recovery (or repair) time/Number of Failures
  - MDT: Downtime/ Number of Failures
- **Related primarily to Heisenbugs**
- **Associated metrics**
  - Coverage: conditional probability of successful restoration given that a failure occurred
- **What requirements must clarify**
  - Definition of recovery time
    - Can be limited only to automated recovery actions or to automated recovery plus manual recovery plus replacement times (for hardware)
    - Example ambiguity: is it from when the failure starts or the failure conditions is diagnosed and recovery process begins?
  - Definition of Downtime
    - Example ambiguity: when is the system really down?
  - Definition of Failure
  - Data collection and analysis process
  - Analytical Procedures

Availability

- **Definition**
  - MTBF/(MBTF+MTTR)
  - MUT/(MDT+MUT)
- **Related primarily to Heisenbugs**
- **What requirements must clarify**
  - See previous issues concerning uptime and downtime
  - Logistics delay (time needed to bring necessary resources and skills to effect the repair)
    - "Inherent availability" (aka "dependability") vs. "Operational availability"
    - Example ambiguity: does the downtime due to logistics delay included in downtime?

Design Life/Mean Mission Duration

- **Definition:**
  - Design Life: Expected value of the life of the system based on the roll up of constituent elements and considering all termination causes (including mission conclusion, random failures, exhaustion of expendables, and wearout)
  - Mean Mission Duration (MMD): Time in orbit at which a satellite will have a 50% chance of failing. This estimate is based on reliability calculations of constituent elements rolled up to the satellite level.
- **Related primarily to Bohrbugs**
  - For software, both design life and MMD are determined by the probability of non-recoverable failures
  - Primarily a quality issue
  - Addressed by development process metrics
- **For safety critical failures, this should be 0.**
  - Addressed by safety program

Problematic Specification Language: Reliability

- **Existing:**
  - "The Space Vehicle (SV) shall have a mean mission duration (MMD) of at least 5.5 years "
- **Problems:**
  - Single value not sufficient: need to specify acceptable probabilities for full service and degraded modes
  - Does not distinguish between recoverable and non-recoverable failures: (most software failures are recoverable)
- **Better:**
  - "The SV shall have a MMD of not less than 5.5 years and a minimum MTBF for each service of not less than [TBD]"

Problematic Specification Language: Maintainability

- **Existing:**
  - The Ground Segment [MTTR] shall be …consistent with the dependability requirement.
  - The Ground Segment design shall utilize a modular or plug-in concept to the greatest extent possible.
- **Problems**
  - Does not address first-level software maintenance requirements:
  - Does not address second-level software maintenance
- **Better:**
  - The ground segment shall have an MTTR of no greater than [TBD] for any hardware or software failure. Monitoring and diagnostics provisions shall detect no less than [TBD, >95%] of all system failures.

Problematic Specification Language: Availability

- **Existing**
  - Ao = Uptime/(Total Time) = MTBDE/(MTBDE + MDT) shall be greater than 0.988
- **Problems**
  - What is a “DE” (downing event)?
  - Over what time interval are these measured?
  - How to distinguish between downing events and other types of failures (e.g., degraded mode)?
  - How to specify availability for degraded modes?
- **Better**
  - The operational availability for each mode shall be as defined in [Table TBD]

Other Specification Issues

- **Failure probability constraints should reflect that some failures more important than others**
  - Non-reversible events (spurious engine start or stop, squib energization, weapons launch)
  - Incorrect data leading to non-reversible events (e.g., corruption of essential non-volatile data)
- **Requirements should address diagnostics, testing, and failure detection probabilities (coverage)**
  - Diagnosis is often more difficult and time consuming than repair and replacement – particularly for software and commercial grade IT equipment
  - Essential for remote systems (e.g., satellites)
- **Failures should be defined with respect to quality of service requirements**
  - Does missing a response time count as a failure? Is there a margin?
  - What is the averaging interval for bit error rate and capacities? What is the definition of a failure for transient surges in BERs or capacity reductions?

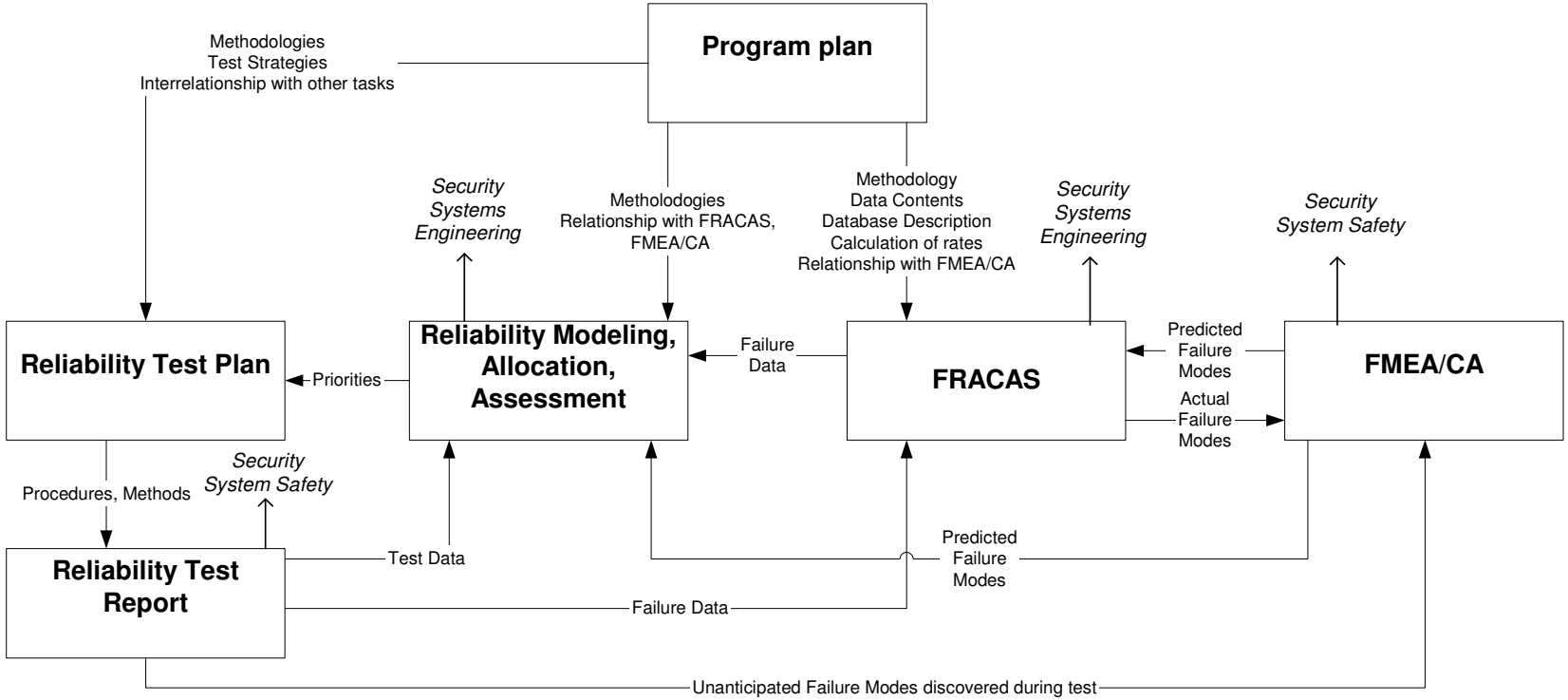
Problems with current DoD standards

- **MIL STD 785B**
  - The authoritative standard on non-spaceborne weapon system reliability
    - Last major revision was more than a quarter of a century ago
    - Cancelled in 1996 in acquisition reform
      - Language in intended replacements, SAE JA 1000 and JA 1002, is not sufficient specific to be contractually enforceable – tailoring is difficult to impractical
  - Oriented almost completely toward hardware
    - Program planning tasks do not adequately address interaction with software development tasks
    - Modeling and prediction tasks do not address software recovery and fault tolerance and their role in availability
    - Failure Reporting and Corrective Action System (FRACAS) and Failure Modes and Effects Analysis (FMEA) tasks do not adequately account for software
    - Testing tasks do not address software reliability (example: collecting processor operating time)

Problems with current DoD standards (continued)

- **MIL STD 1543B**
  - Standard based on MIL STD 785B for space and launch vehicles
  - Last updated almost two decades ago
  - Greater recognition of software, but shares deficiencies with respect to modeling, prediction, and failure reporting
- **Associated Data Item Descriptions**
  - Most not updated for 20 years
  - Contents do not adequately address software
  - Many ambiguities
  - Obsolete hardcopy concept that affects data content and delivery
    - Example: FMEAs and failure reports should be submitted in XML
    - Discrete deliveries should be supplemented by continuous updates (subject to configuration management constraints)

Example Reliability Program Plan for Hardware and Software



Reliability/Availability Modeling and Allocation Tasks

- **Include both hardware and software**
  - Account for software failures and recoveries
  - Include recovery times and recovery probabilities (for automated failure detection and recovery)
  - For software, recovery times and recovery probabilities have as large an impact as failure rates
- **Model re-used (commercial or previously developed) software with operational history or data derived from observations**
  - Generally feasible to create test systems that reasonably simulate the run-time environment (workload, error rates, etc.)
  - Perform failure/recovery testing
- **Allocate parameters for developed software**

Failure Modes and Effects Analyses (FMEA)

- **Include software failure modes at functional level**
- **Failure mode definitions should be based on run-time objects (tasks, processes), not pre-compilation elements (modules, packages, libraries)**
- **Taxonomy of failure modes and effects based on discrepancy reports during software development**
  - Crash
  - Hang
  - Stop
  - No response
  - Incorrect response
  - Late/early response
- **Monitor actual failures during integration testing to establish whether there are unanticipated failure modes or effects**

Failure Reporting and Corrective Action System (FRACAS)

- **Utilize collected data to establish software reliability/availability**
  - Failure rates, recovery times, recovery probabilities, confidence intervals
  - Qualitative information (symptoms, causes, diagnostics and recovery effectiveness)
- **Necessary data to collect**
  - Operating time
  - All failures (not just unique or reproducible)
  - Symptoms and causes
  - Detection methods
  - Recovery times
  - Utilize automated system logs
- **Feedback to analyses**
  - Failure rates, recovery times, and recovery probabilities to models
  - Failure modes, diagnostics effectiveness, and causes to FMEA
  - Causes to software development and hardware development

Conclusions

- **Reliability and availability are essential quality attributes to end users and customers**
- **Past practices are based on sound principles**
  - Planning,
  - Allocation
  - Prediction
  - Data collection, and feedback
- **Need to be updated to reflect software as a contributor to failure behavior**

Acronyms

DID: Data Item Description
DoD: Department of Defense
FAR: Federal Acquisition Regulations
FMEA: Failure Modes and Effects Analysis
FRACAS: Failure Reporting and Corrective Action Systems
NASA: National Aeronautics and Space Administration
RMA: Reliability, Maintainability, Availability