LOCKHEED MARTIN

INTEGRATED VEHICLE
HEALTH MANAGEMENT

Development of a Flight-Critical Software Failure Taxonomy

Walter A. Storm
Jung N. Riecks

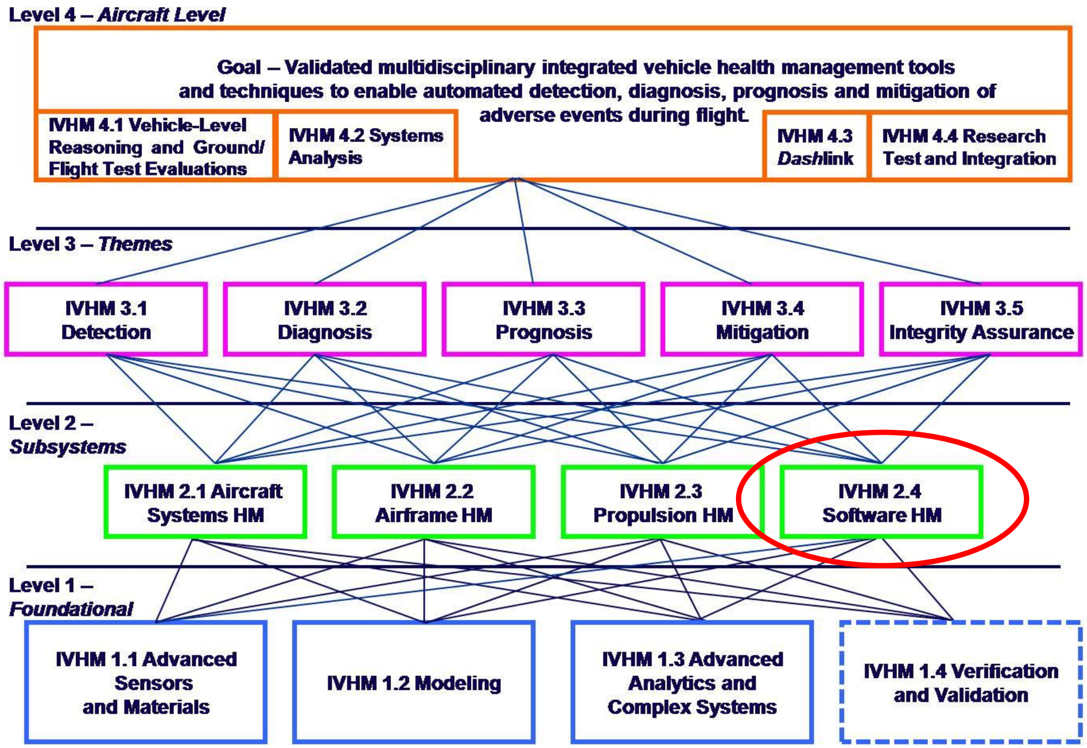
DISTRIBUTION STATEMENT A. Approved for Public Release REF. LM-Aero PIRA: AER200905008.

Aviation Safety Program Technical Conference
November 17-19, 2009
Washington D.C.

Outline

LOCKHEED MARTIN

- Problem Statement
- Background
- IVHM milestones(s) being addressed
- Approach
- Results
- Conclusions
- Future Plans



Problem Statement

NASA's Aviation Safety (AvSAFE) Program's Integrated Vehicle Health Management (IVHM) project has identified the need for foundational research that will enable the development of technologies for automated detection, diagnosis, prognostics, and mitigation of adverse events due to aircraft software, and is exploring software health management in the context of system level dependability cases[1]

Problems being addressed in this effort:

- What are the anticipated flight–critical software failure modes
- Can suitable abstractions of these failure modes be developed for software health management purposes; and if so,
- Is it possible to use these failure mode abstractions for prioritizing risk

1. Jackson, D.; et al. *Software for Dependable Systems: Sufficient Evidence?* National Academies Press, 2007

Background

IVHM is exploring software health management in the context of system level dependability cases by developing a framework that enables:

- Explicit claims of system (and subsystem) requirements including assumptions about the application domain and environment in which the system is to operate
- Evidence that software satisfies these explicit claims under the stated domain assumptions
- Architectural principles, enforced by hardware mechanisms, that ensure that software behavior dependencies are traceable; and
- Mechanisms for correctly composing software systems from trusted components within the constraints imposed by the architectural principles

IVHM milestones(s) being worked

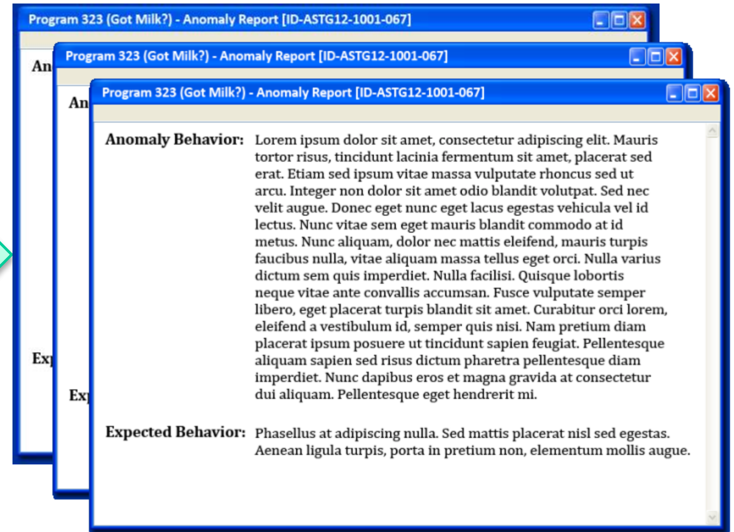
- **Milestone 2.4.5.2** Framework for accumulating evidence that observed behavior, including both inputs and outputs, of a software system is consistent with its expected behavior.
  - Metric i) Perform a study to catalog historical aircraft software anomalies to include representative anomalies uncovered during pre-deployment verification and validation activities as well as those discovered post-deployment. From this catalog a set of working metrics will be derived for developing an evidence base.

Approach

- Create a taxonomy of failure types for flight-critical software systems.
- Create useful abstractions of failure classes.
- Analyze the data to identify high-risk error classes and error types.
- Use the taxonomy to suggest approaches to anticipate and address errors.

Preparation

Program X
Program Y
Program Z
Program α

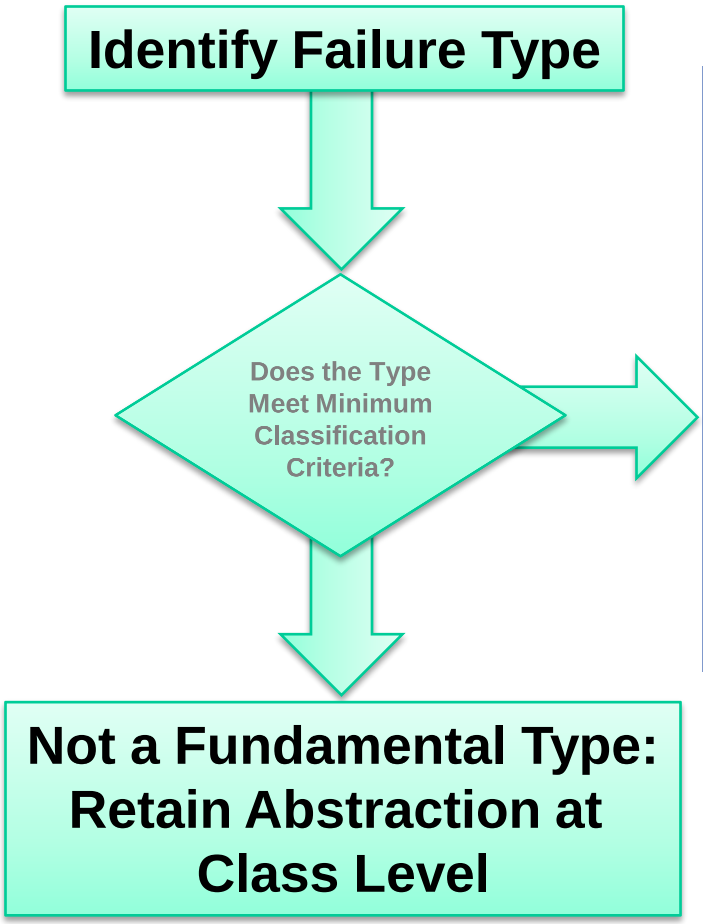


Identify Failure Class

Identify Failure Type

Taxonomy Development Process

LOCKHEED MARTIN



Algorithm Failure Class (Cont'd)

Failure Type	Definition
missing initialization	missing initialization function
missing limiter	missing limiter in the calculation
prototype	missing prototype
range	incorrect or unnecessary range in calculation or condition
relational operator	incorrect relational operator (i.e. >, <, >=, <= ...)
reset logic	incorrect reset algorithm
reset timing	incorrect reset timing
response to detected failure condition	incorrect repose to detected failure condition
sampling time	incorrect sampling time
setting value/variable	incorrect algorithm to setting values or variables
syntax	syntax error
test modeling	incorrect test modeling produce incorrect values for the test
threshold	incorrect threshold
timing	incorrect delay
typo	typo in algorithm causes disconnect between signals
validity check timing	missing or incorrect or inappropriate timing of validity check

Results

LOCKHEED MARTIN

- After several passes through the data by various subject matter experts, the LM Aero team converged on a comprehensive failure taxonomy consisting of:
  - 16 Fundamental Failure Classes
  - 114 Fundamental Failure Types

Fundamental Failure Classes
Algorithm
Bus Interface
Compiler Error
Configuration Management
Data Definition
Data Handling
Documentation
Hardware
I/O system
Implementation
Inter-Process Communication
Performance
Self-Test
System Integration
Tools
User

Sample Taxonomy Entries

LOCKHEED MARTIN

Bus Interface Failure Class	
Failure Type	**Definition**
bit position	incorrect bit position
bus initialization failure	bus initialization failure
data source	incorrect data source is connected to bus interface
missing signal	missing a signal in bus interface

Configuration Management Failure Class	
Failure Type	**Definition**
approval delay	correct version of SW was not approved.
implementation delay	implementation not incorporated into latest build configuration
incorrect version of software	using incorrect version of SW
missing CR implementation	missing CR implementation
outdated requirement	did not update requirement to match a SW change
requirement incorporation delay	did not update SW to match a requirement change

Compiler Error Failure Class	
Failure Type	**Definition**
Incorrect Assembly Code	Incorrect Assembly Code

Conclusions

LOCKHEED MARTIN

- **The Risk Priority Number (RPN)**
  - **A normalized value, between 0 and 1000, that indicates the overall risk of an error class or type.**

$$RPN = O \times S \times D$$

Where:

$O := Relative\ Frequency\ of\ Occurance$

$S := Severity\ of\ Error$

$D := Phase_{Detected} - Phase_{Injected}$

Severity	weight
1	10
2	8
3	5
4	2
5	1

Normalization Table for Severity

Normalization Table for Delta-Phase

	Defect Detection Phase						
Defect Introduction Phase	Planning	Requirements	Design	Code	Integration and Test	Transition to Customer	Fielded Defect
Planning	1	2	4	6	9	10	10
Requirements		1	3	5	8	10	10
Design			1	4	7	10	10
Code				1	6	10	10
Integration and Test					1	10	10
	Weight Factor						

Prescriptions of the RPN Model

LOCKHEED MARTIN

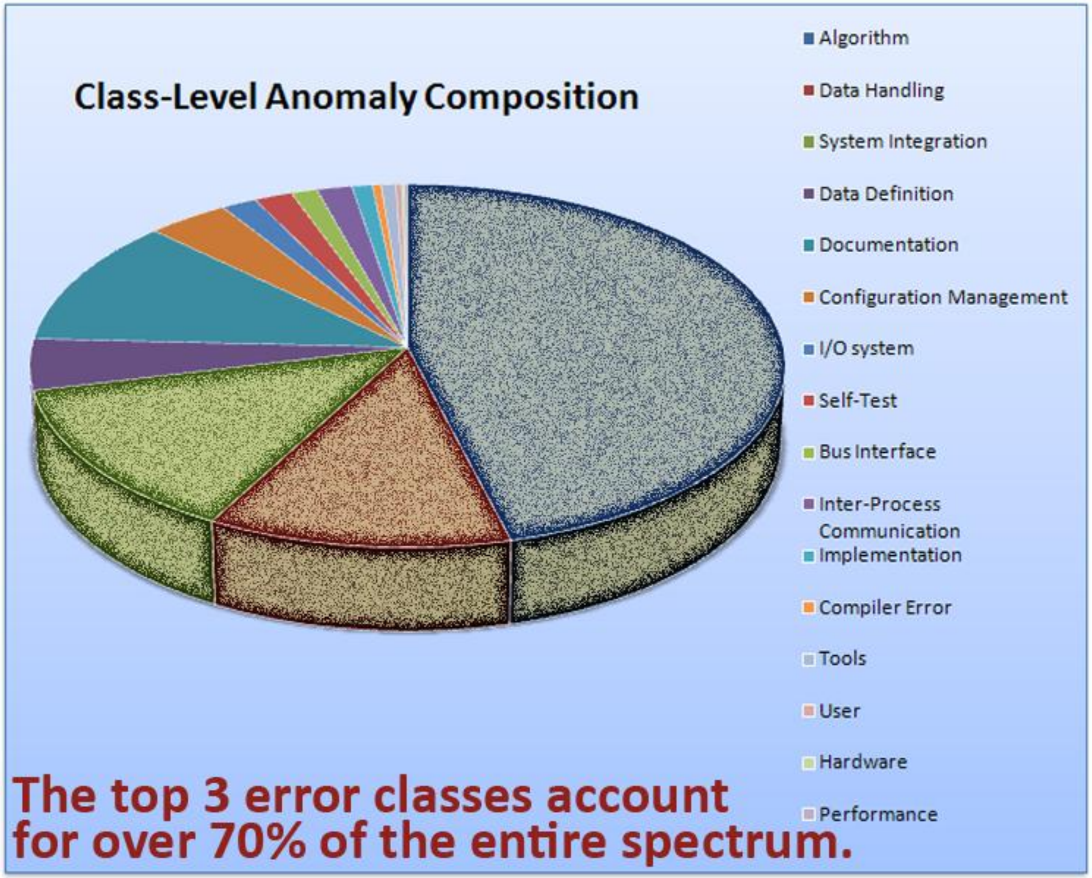
- **Any element with an RPN over 100 is generally considered *high-risk*.**
- **Such elements require mitigation schemes such as system health monitors or formal design verification.**

Error Class	Error Type	RPN
Algorithm	design	774
Algorithm	decision logic	353
Algorithm	data transfer/message	350
Data handling	scaling factor	324
Documentation	Documentation error	262
Algorithm	failure management	228
Algorithm	reset logic	203
Data handling	memory address	188
Algorithm	initialization of values	169
Algorithm	failure isolation	133
System Integration	incorrect requirement	127
Algorithm	setting value/variable	120
Algorithm	initialization logic	119
Algorithm	timing	113
Algorithm	range	113
System Integration	no requirement	105

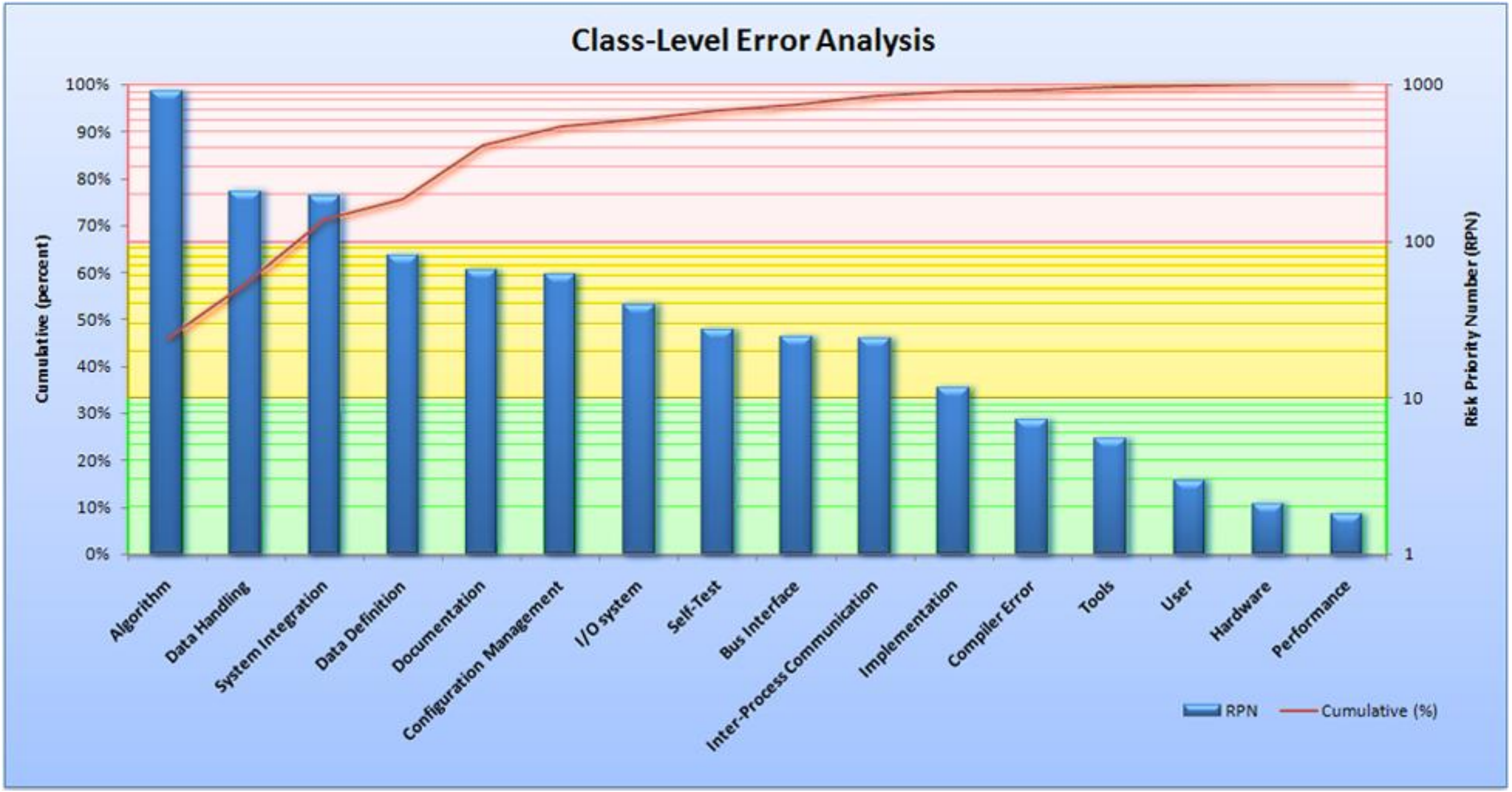
Class-Level Results

LOCKHEED MARTIN

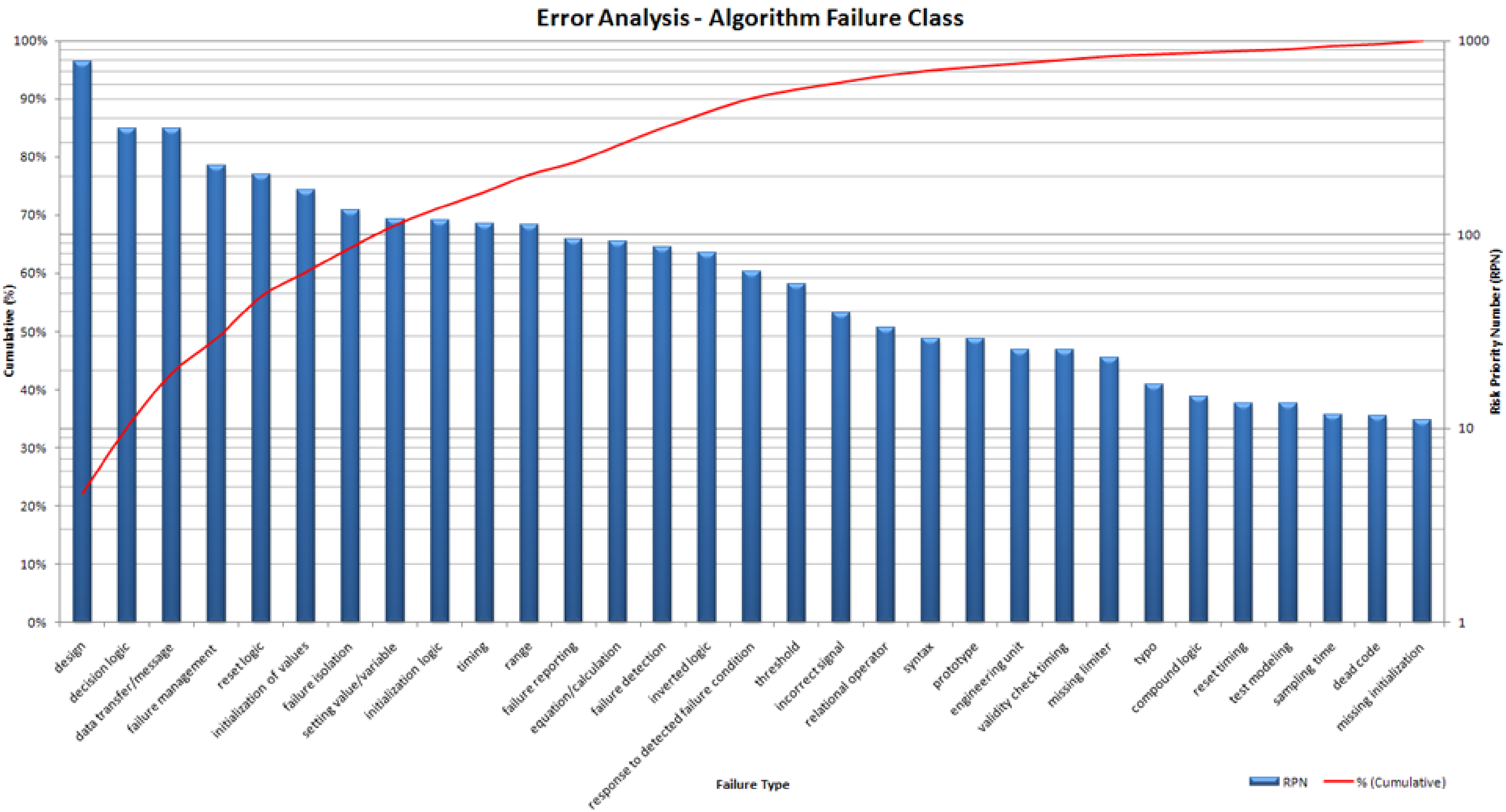
- **Algorithm, Data Handling and System Integration Errors combined account for over 70% of all error types.**



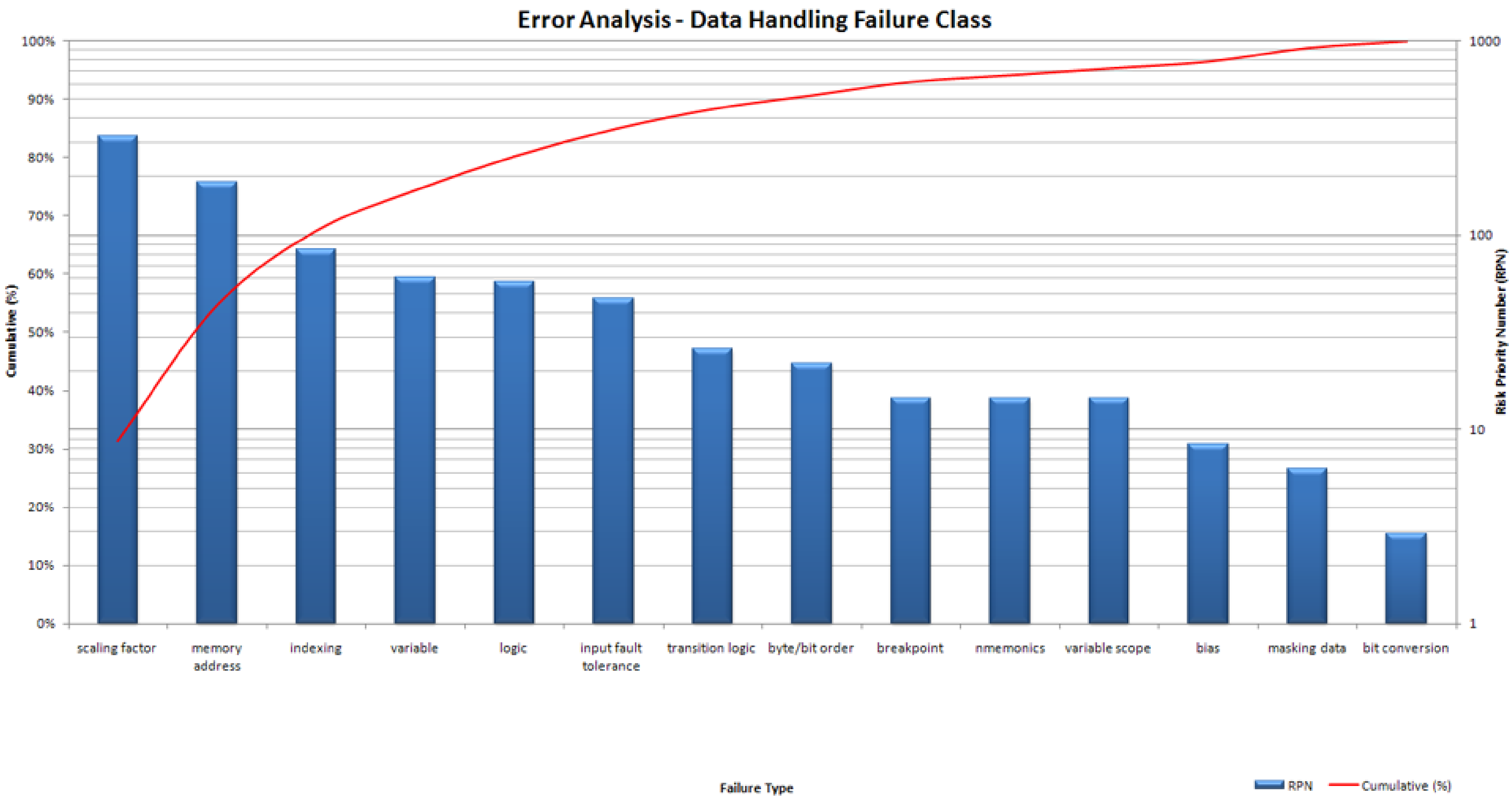
Class-Level Results



Algorithm Results

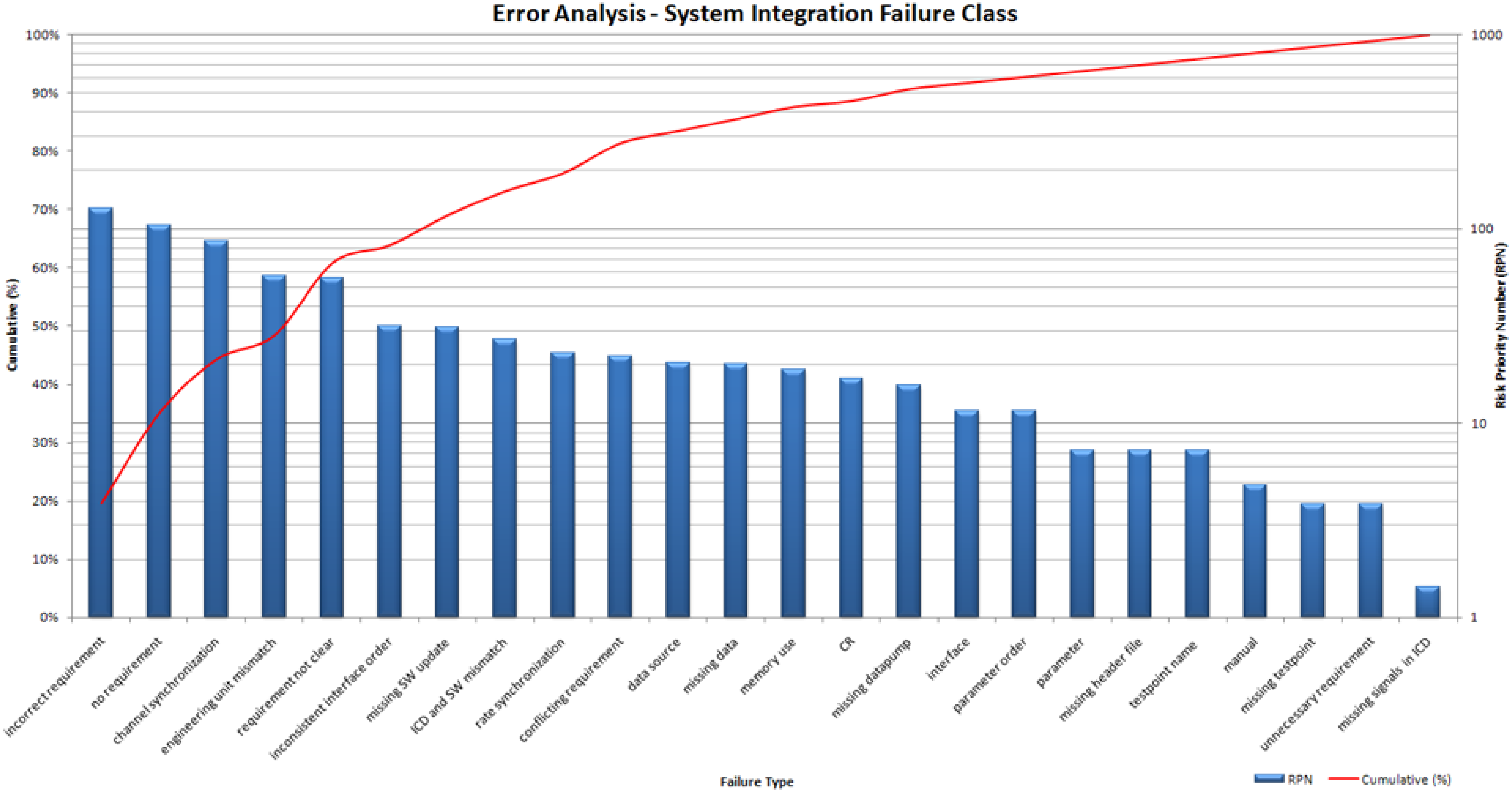


Data Handling Results



System Integration Results

LOCKHEED MARTIN



Next Steps

LOCKHEED MARTIN

- Validate the Taxonomy.

- Expand the details of the causal analysis.

- Use this information to seed algorithms that detect and react accordingly in the context of a software health management system.

LOCKHEED MARTIN