

---

# ***Software Dependability Techniques***

## A b s t r a c t

---

*This module presents techniques, methods and procedures of analysis, definition and verification of requirements related to software Dependability and Safety.*

# Space missions lost due to software failures



- **Mariner 1 (July 1962)**
  - omission of a hyphen in a computer instruction
- **Ariane 501 (June 1996)**
  - unhandled software exception
- **Titan IV-B (April 1999)**
  - incorrect parameter in the attitude control system
- **Mars Climate Orbiter (September 1999)**
  - failed translation of English units into metric units

## Other recent failures

- **Mars Polar Lander**
- **Deep Space 2 (December 1999)**
  - ☞ cause uncertain
  - ... but the testing activity was found not adequate

# Space missions lost due to software failures

**“What are the lessons learned from these failures?”**

***A thorough verification program is essential for mission success.***

*from “Mars Program Independent Assessment Team Summary Report”, NASA, 14.03.2000*

- Since Mariner 1 failure in 1962, no missions have been lost because of documented software failures until Ariane 501 in 1996
- Between 1996 and 1999 several missions have been lost because of software failures

## **WHY?**

**Software plays more and more a central role in space (and non-space) systems,  
therefore  
its dependability and safety shall be carefully analysed**

# R . A . M . S . = D e p e n d a b i l i t y + S a f e t y

---

- The “root” cause of SW failures is the lack of appropriate specification, implementation or verification of **SW RAMS** requirements
  - Reliability:** continuity of service.*
  - Availability:** readiness for usage.*
  - Maintainability:** easiness of repair/upgrade.*
  - Safety:** non-occurrence of catastrophic failure*
- Appropriate techniques, methods and procedures shall be adopted in order to:
  - ensure that **SW RAMS** requirements are specified and taken into account during the design and implementation of software systems
  - verify and validate that the implemented SW systems comply with the specified **SW RAMS** requirements

## R. A. M. S. in the ECSS serie

---

### Q-30 ECSS-Q-30 Space product assurance. Dependability

- ✧ it defines the requirements for a dependability (**reliability, availability and maintainability**) assurance programme for space projects.

### Q-40 ECSS-Q-40 Space product assurance. Safety

- ✧ it defines the safety programme and the technical **safety** requirements for space projects.

### Q-80 ECSS-Q-80 Space product assurance. Software product assurance

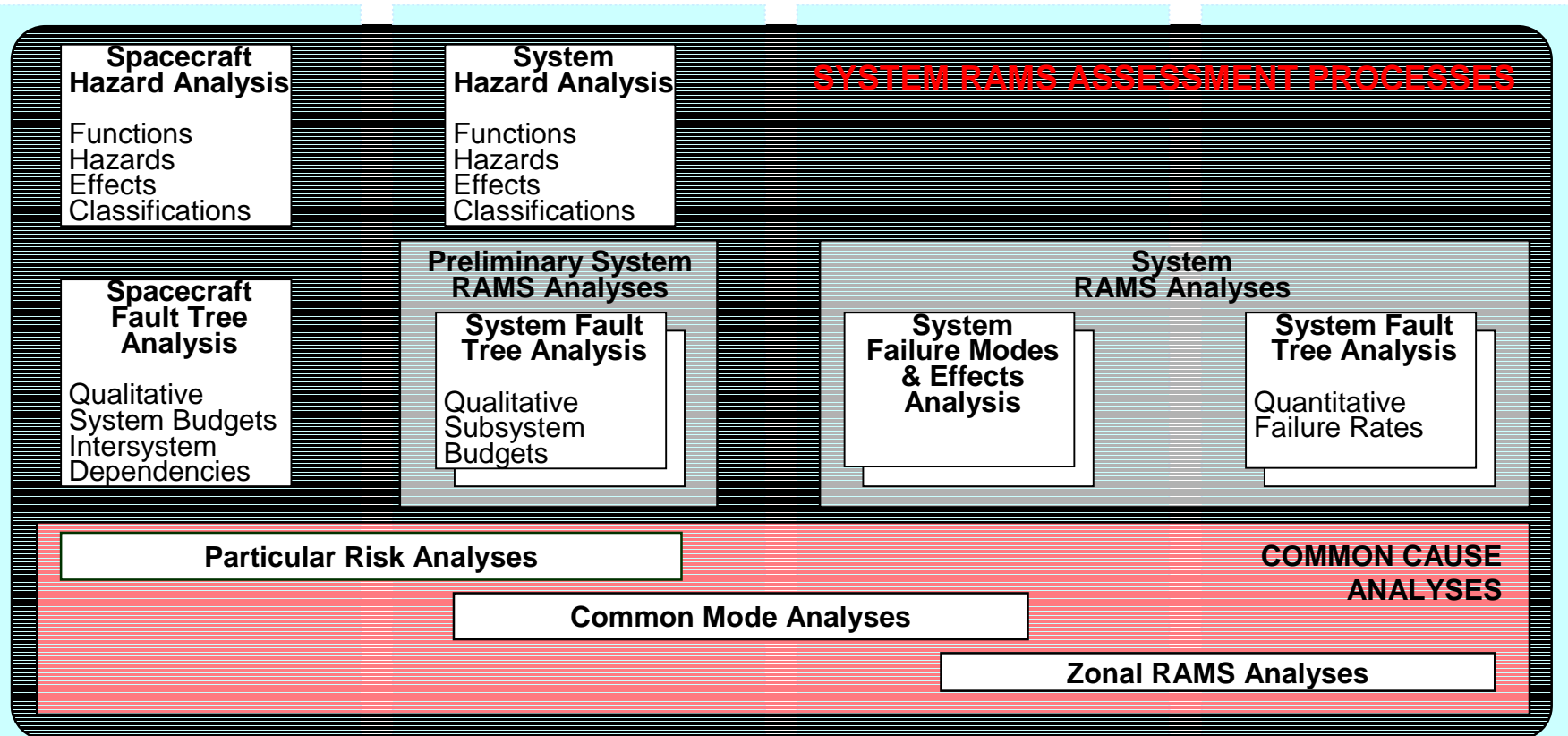
- the requirements addressing Software Dependability and Safety are specifically defined :
  - *6.2.2 Software dependability and safety analysis*
  - *6.2.3 Handling of critical components*
- they need to be tailored for Dependability and Safety aspects according to the specific criticality categories.

## System vs. Software RAMS

---

- Software implements a large part of space systems functionality
  - the System RAMS approach needs to be supported through correspondent **Software RAMS processes**
  - **Software RAMS requirements** need to be derived from system RAMS recommendations
- System RAMS analysis at functional level is used to identify the critical functions
  - Software RAMS is to identify typical **software failures modes** (e.g. deadlock, task overrun, buffer overflow, divide by zero).
  - Software RAMS requirements need to be specified to ensure fault tolerance (e.g. through FDIR, watch-dog, exception handling, etc.) and operational contingency

# System RAMS Analysis



**Concept Development**

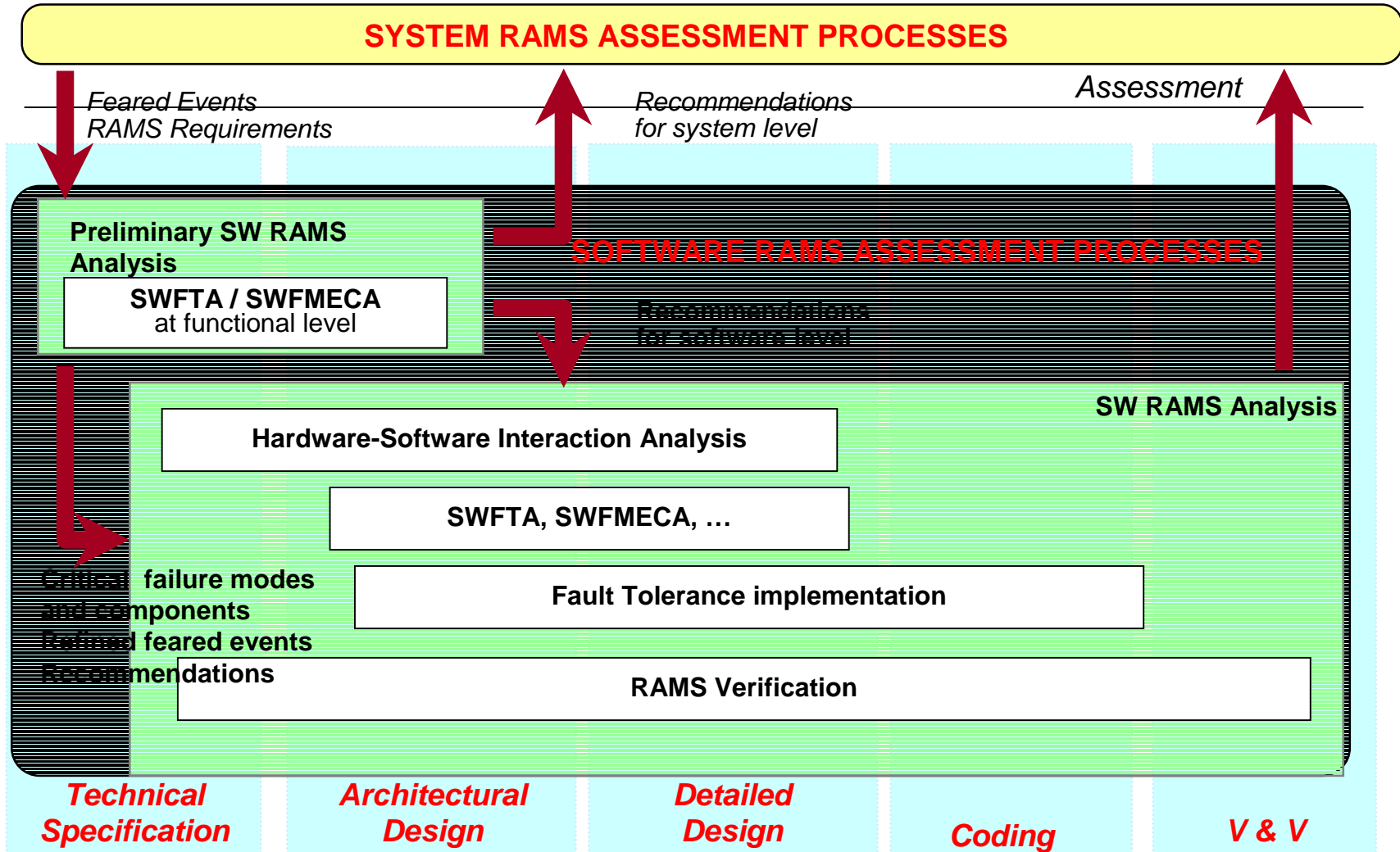
**Preliminary Design**

**Detailed Design**

**Design V & V**

from: "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment" - SAE Aerospace Recommended Practice, ARP4761, 1996

# Software RAMS Analysis



# Software Failures and Faults

---

## Error

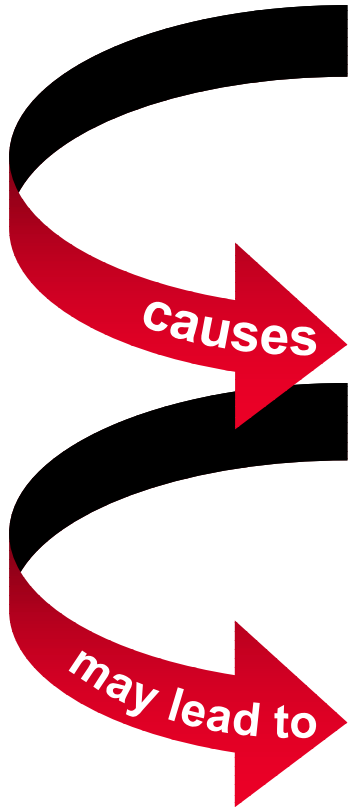
**A wrong or missing human action or thought made while trying to understand given information, to solve problems, or to use methods and tools.**

## Fault

**An incorrect step, process or data definition in a program.**

## Failure

**The inability of the software to perform its required functions.**



*IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, February 1991*

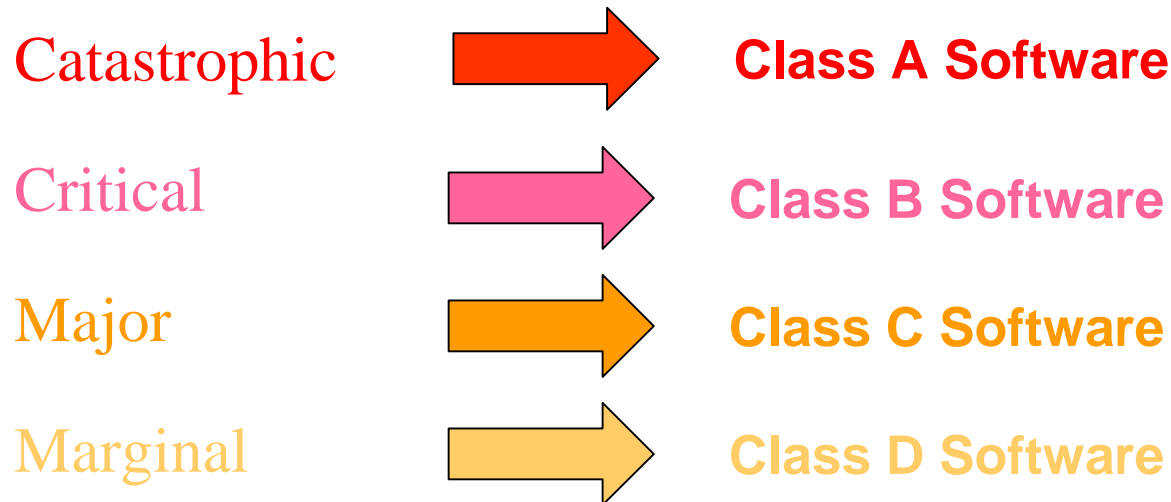
## Software Failures Consequence and Severity Category

---

- Software may cause directly, indirectly or contribute to cause, in association with other sources (operators, hardware), events which are undesirable.
- The degree of severity consequences is a mean to classify those events in categories.
  - Catastrophic
  - Critical
  - Major
  - Marginal

## Sw Severity Categories vs. Sw Criticality Classes

- A one to one correspondence is not always feasible
  - ✓ A possible correspondence may be



Q-80B

6.2.2.1

A functional analysis at system level shall be used to identify the critical software components (see ECSS-Q-30 and ECSS-Q-40)...

## RAMS activities

---

- Software RAMS supports the following approaches:
  - **Fault Prevention**
    - *avoidance/reduction of fault causes*
  - **Fault Tolerance**
    - *avoidance/reduction of fault consequences*
  - **Fault Removal**
    - *removal of fault occurrences*
  - **Fault Forecasting**
    - *prediction of behaviour in presence of faults*

# RAMS activities

## et Fault Prevention

- ✓ Hazard Analysis
  - analysis at system/software level
- ✓ Architectural Design
  - specific design choices to prevent faults
- ✓ Implementation
  - implementation of specific functions

## et Fault Tolerance

- ✓ Architectural Design
  - specific design choices to tolerate faults
- ✓ Implementation
  - implementation of handlers

## Production activities

## et Fault Removal

- ✓ Testing
  - functional and performance testing activities
- ✓ Inspection
  - analysis of products

## et Fault Forecasting

- ✓ Reliability evaluation
  - statistical testing and reliability estimation

## Verification activities

## Static Verification Methods, Techniques and Procedures

---

Static activities, which do not require the execution of the software

### Hazard Analysis

- Sw Fault Tree Analysis
- Sw Failure Mode Effect & Criticality Analysis
- Hw/Sw Interaction Analysis
- Sw Common Cause/Mode Failure Analysis
- Sw Error Effect Analysis

### Concurrent Behaviour Analysis

- Specification Description Language (SDL)
- Schedulability Analysis
- Finite State Machines (FSM)

### Product Analysis

- Code Analysis
- Inspections
- Walk-Through
- Software Sneak Analysis

### Numeric Analysis

- Computational Accuracy

## Dynamic Verification Methods, Techniques and Procedures

Dynamic activities, which require the execution of the software

### White Box Testing

Statement Coverage  
Branch Coverage  
Path Coverage  
Basis Path Coverage  
Multiple Condition Coverage  
Fault Injection

### Test Data Selection

Boundary Value Analysis  
Equivalence Partitioning

### Black Box Testing

Back-to-Back Testing  
Interface Testing  
Stress Testing

### Test Analysis

Test Result Analysis  
Test Coverage Analysis

### Regression Analysis

## Design Constraints

---

- A number of **Design & Coding Practices** can be applied in order to
  - adopt specific architectural choices to prevent or tolerate faults
  - implement specific functions to prevent faults
  - implement specific recovery actions to tolerate faults

### Design & Coding Practices

- Defensive Programming
- N-Version Programming
- Segregation/Partitioning
- Watchdog

## Organisational Constraints

---

- Some RAMS constraints force the adoption of specific rules and procedures for the development organisation and/or development process
- A number of **Organisational Practices** can be applied in order to
  - enhance the production process, and
  - prevent the introduction of faults

### Organisational Practices

- Independent V&V
- Traceability Analysis
- Certified/Qualified/Validated Tools

## Summary

**The root cause of software failures is the lack of appropriate specification, implementation or verification/validation of software RAMS requirements.**

- **Software RAMS requirements** (Safety & Dependability) need to be:
  - specified in the technical specification definition process,
  - considered in the software design, implementation and verification/validation processes.
- Appropriate **Software RAMS practices** allow to:
  - verify the implementation of software RAMS requirements
  - mitigate the safety & dependability risks.
- Different Software RAMS practices address different **RAMS measures**:
  - Fault Prevention and Tolerance during Production
  - Fault Removal and Forecasting during Verification

## Summary : Recommended Sw RAMS Practices

---

- System RAMS analysis needs to be consistently supported through dedicated **Software RAMS analysis** (*Static Verification*)
  - **Hazard Analysis** (SFTA, SFMECA, HSIA, SCCFA)
  - **Behavior Analysis** (SDL, Petri Nets, Schedulability Analysis)
- **Software RAMS requirements verification** can be achieved through the proper use of testing techniques (*Dynamic Verification*)
  - **White Box Testing** (Coverage Testing, Fault Injection)
  - **Black Box Testing** (Back-to-Back Testing, Stress Testing)
  - **Test Data Selection** (Boundary Value Analysis, Equivalence Partitioning)
- Software RAMS properties can be enhanced through the adoption of **Design** and/or **Organisational practices**
  - e.g. Safety Supervisor, Segregation/Partitioning, N-Version Programming and others
  - e.g. Independent Verification and Validation and others

## Conclusion

- Specific SW RAMS requirements shall be based on the overall system RAMS
- SW RAMS analysis shall be applied in order to prevent possible faults
- SW RAMS requirements verification shall be applied in order to remove (and possibly forecast) possible faults
- Specific design constraints shall be applied in order to prevent and tolerate possible faults
- Specific organisational constraints shall be applied in order to increase the confidence in the software product
- SW RAMS practices shall be systematically applied for treatment processes in order to eliminate critical areas identified through RAMS analysis

