Developing and Evaluating Critical Software for Space Systems

the Do’s and the Don’ts

Lothar Winzer

Head of Software Product Assurance Section

ESA/ESTEC Product Assurance and Safety Department

Apr-17-09

List of past “new challenges” or “new promises”

- Which have been overcome
- Which are “work in progress”
- For which are we pretty much still where we were despite trying hard? Why?
- Which are not in the list, new?

Outline

1. Motivation
   - ➢ Space Mission Failures
2. Past “new challenges”
   - ➢ Software Process Assessment and Improvement
   - ➢ Evaluation of Existing Software for Reuse
   - ➢ Independent (third Party) Verification and Validation
3. “Work in Progress”
   - ➢ Software Quality Modeling and Metrication
   - ➢ Use of COTS SW & OSS in Critical Functions
   - ➢ Model based Validation of RAMS Requirements
4. For which are we pretty much still where we were
   - ➢ RAMS at Software Level
   - ➢ “Certification” of Software Products
   - ➢ Risk Management using Process Assessment Results
5. Which are not in the list
   - ➢ Software Reliability Modelling
   - ➢ N-version programming

1. Motivation: Space Missions Failures

Mission	Failures
Ariane 501	The launcher veered off its flight path, broke up, and exploded.
SOlar Heliospheric Observatory (SOHO)	Errors in performing the calibration and momentum management manoeuvre and in recovering from an emergency saving mode led to the loss of telemetry.
Mars Polar Lander (MPL)	A problem occurred during the entry, deployment, and landing (EDL) sequence when the three landing legs were to be deployed from their stowed condition to the landed position.
Titan IV B-32	The Centaur lost all attitude control. The Centaur went into a very low orbit and the Milstar satellite separated from the Centaur in a useless final orbit.
Mars Climate Orbiter (MCO)	The MCO was lost when entered the Martian atmosphere in a lower than expected trajectory.

Causes of Mission Flaws (I)

Ariane 501 (June 1996)

- Unhandled Software exception.
  - The Software was reused from the Ariane 4 and included functions that were not needed for Ariane 5 and were left in for having common characteristics.
  - The requirements and specifications did not include the trajectory data. An overflow of a 16-bit integer variable occurred.
  - The Software did not include simple and fairly standard range and overflow checks.

SOlar Heliospheric Observatory (June 1998)

- A series of errors were introduced in making the Software changes.
  - Limited Software reviews were done to the changes to the ground-generated commands.

Titan IV-B (April 1999)

- Incorrect parameter in the attitude control system.
  - The Software with the load tape prior to launch was not tested.
  - Mission operations personnel did not understand the system or Software well enough to interpret the data they saw as indicating there was a problem in time to prevent the loss.

Source: [RED-1]

Causes of Mission Flaws (II)

➢ Mars Polar Lander (January 1999)

- ✧ The Software interpreted the spurious signals generated at leg deployment as valid touchdown events.
  - Software specifications described the nominal behaviour well, but they were very incomplete with respect to required Software behaviour under off-nominal conditions (did not describe what the Software was not supposed to do).
  - Software designers did not include any mechanisms to protect against transient sensor signals nor did they think they had to test for transient conditions.

➢ Mars Climate Orbiter (September 1999)

- ✧ Failed translation of English units into SI units.
  - There was inadequate identification of mission critical elements throughout the mission. The criticality of specific elements of the ground Software that impacted the navigation trajectory was not identified.
  - No evidence of Software review process.

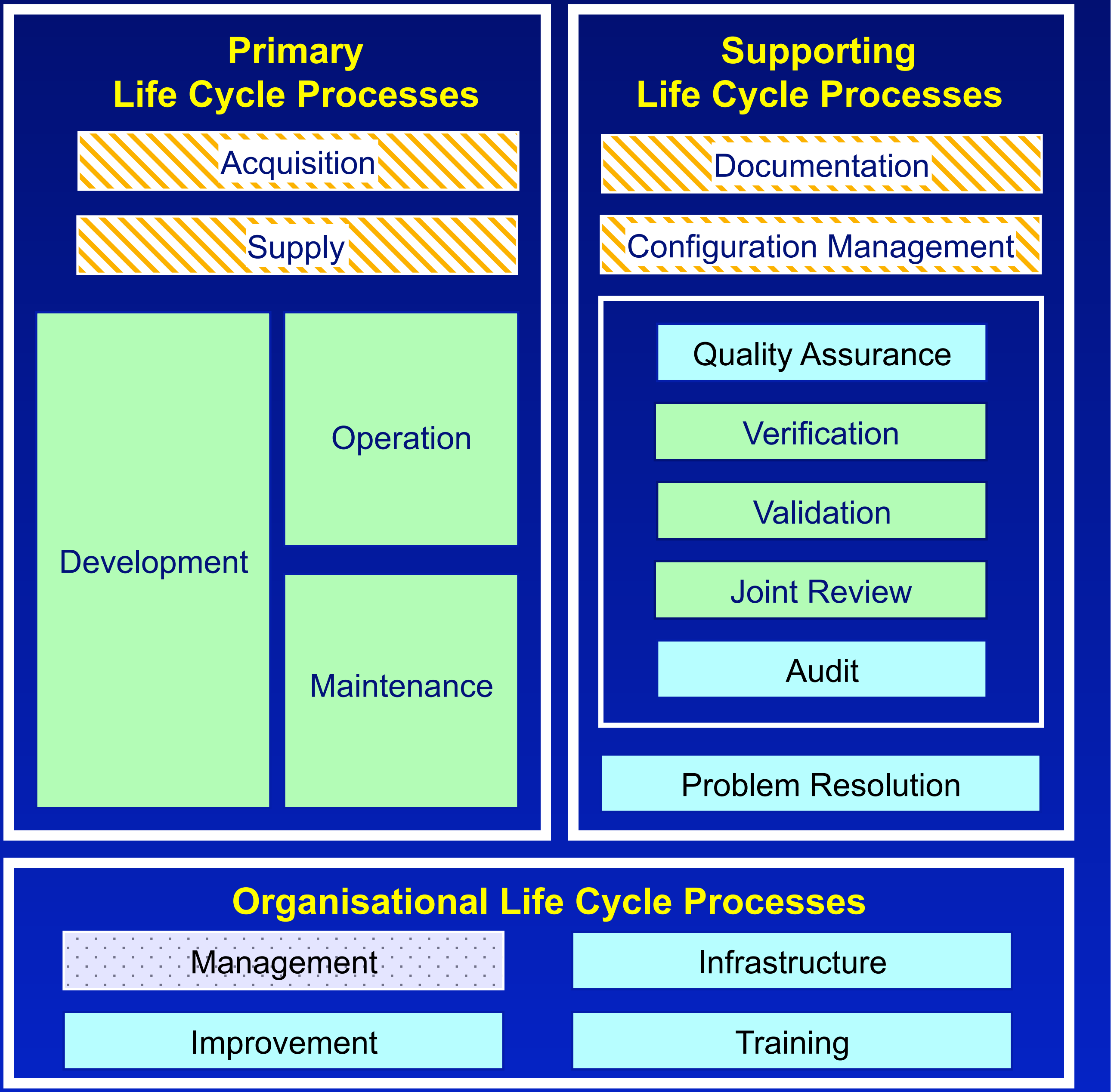
Source: [RED-1]

1. Motivation: Missions Conclusions

- Inadequate system and Software engineering (poor specification practices, unnecessary complexity and Software functions, Software reuse without appropriate RAMS analysis, violation of basic RAMS engineering design practices in the digital components, inappropriate modelling methodologies and analysis tools, etc).
- Ineffective RAMS engineering. Not all the missions defined a RAMS programme/plan.
- Overconfidence and complacency affected the missions activities.
- There was a lack of adequate risk identification, communication, management, and mitigation, which compromised the missions success.
- Inadequate review activities or the absence of them.
- Inadequate human factors engineering.
- Flaws in the test and in the simulation environments.

ECSS Covers the ISO/IEC 12207 Standard

Other ECSS	▨
E-40B	🟩
Q-80B	🟦
Details for SPA and/or SWE	⬚



Primary Life Cycle Processes

Acquisition

Supply

Development

Operation

Maintenance

Supporting Life Cycle Processes

Documentation

Configuration Management

Quality Assurance

Verification

Validation

Joint Review

Audit

Problem Resolution

Organisational Life Cycle Processes

Management

Infrastructure

Improvement

Training

2.1 Past “new challenges”: Process Assessment and Improvement

Primary Life Cycle Processes

- CUS.1.1 Acquisition Preparation
- CUS.1.2 Supplier Selection
- CUS.1.3 Supplier Monitoring
- CUS.1.4 Customer Acceptance
- CUS.2.1 Supply Preparation
- CUS.2.2 Delivery
- CUS.3 Requirements Elicitation
- CUS.4.1 Operational Use
- CUS.4.2 Customer Support
- CUS.5 Contract Maintenance

- ENG.1.1 System Requirements Analysis and Design
- ENG.1.2 Software Requirements Analysis
- ENG.1.3 Software Design
- ENG.1.4 Software Construction
- ENG.1.5 Software Integration
- ENG.1.6 Software Testing
- ENG.1.7 System Integration and Testing
- ENG.2.1 System and Software Maintenance

Supporting Life Cycle Processes

- SUP.1 Documentation
- SUP.2 Configuration Management
- SUP.3 Quality Assurance
- SUP.4 Verification
- SUP.5 Validation
- SUP.6 Joint Review
- SUP.7 Audit
- SUP.8 Problem Resolution
- SUP.9 Safety and Dependability
- SUP.10 ISVV

Organisational Life Cycle Processes

- MAN.1 Management
- MAN.2 Project Management
- MAN.3 Quality Management
- MAN.4 Risk Management
- MAN.5 Information Management

- ORG.1 Organisational Alignment
- ORG.2.1 Process Establishment
- ORG.2.2 Process Assessment
- ORG.2.3 Process Improvement
- ORG.3 Human Resource Management
- ORG.4 Infrastructure
- ORG.5 Measurement
- ORG.6 Reuse

2.1 S4S: ISO 15504 compliant assessment model and method

In comparison with ISO 15504, S4S has:

- **4 New Processes**

✧ CUS.5	Contract Maintenance
✧ MAN.5	Information Management
✧ SUP.9	Safety and Dependability
✧ SUP.10	Independent Software Verification and Validation

- **2 New Component Processes**

  CUS 2.0 Supply ⇒ CUS.2.1 Supply Preparation + CUS.2.2 Delivery

- **Many lower level elements**

  (41 New Base Practices, 73 New Notes, 22 New Work Products)

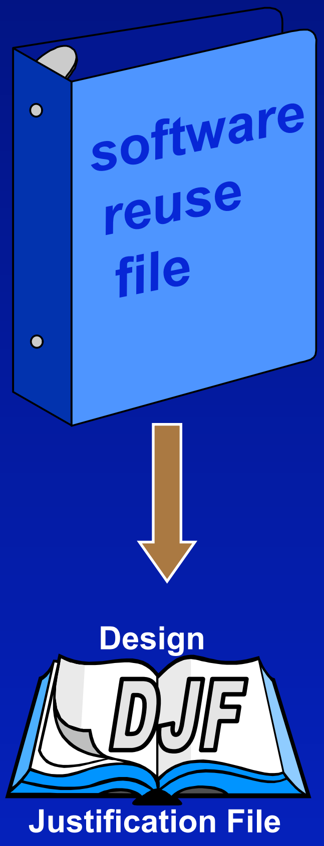
Note: CMMI is widely used specifically by big companies with large US customer base

Source: [RED-1]

2.2 Past “new Challenges: Evaluation of Existing Software for Reuse (1)

Considerations in the choice of reused software include:

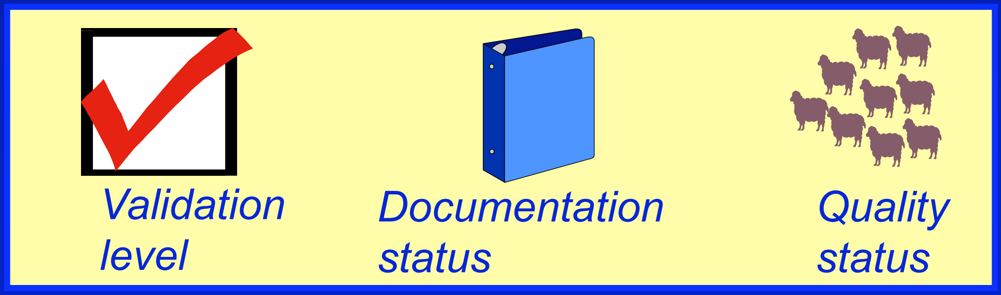
- assessment of the reused software / product with respect to all applicable requirements (including quality requirements)
- criticality of the function to be provided
- acceptance and warranty conditions
- available support documentation
- conditions of installation, preparation, training and use
- Identification/registration by CM
- maintenance responsibility
- maintenance conditions, including possibilities for changes
- copyright/modification rights & license conditions.

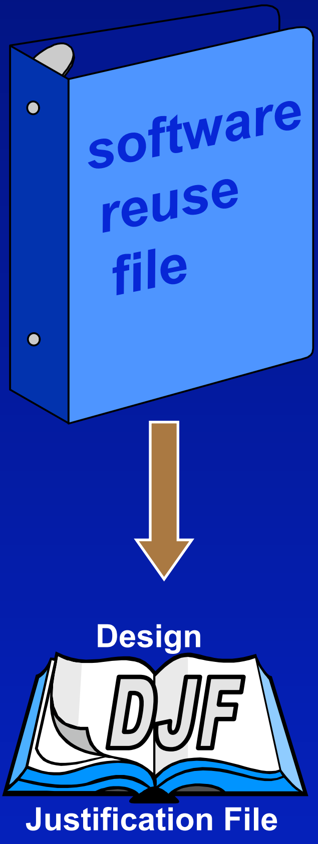


Source: [RED-1]

2.2 Past “new Challenges: Evaluation of Existing Software for Reuse (2)

- **After selection of reused software, it is necessary to check certain aspects and define any necessary corrective actions**
  - ✧ Methods and tools used in the original development must be analyzed for their continuing durability and validity
- **For each reused component, the following must be checked:**
  - ✧ validation level or operational behaviour
  - ✧ documentation status
  - ✧ quality status (residual nonconformances, complexity analyses, or waivers)





Source: [RED-1]

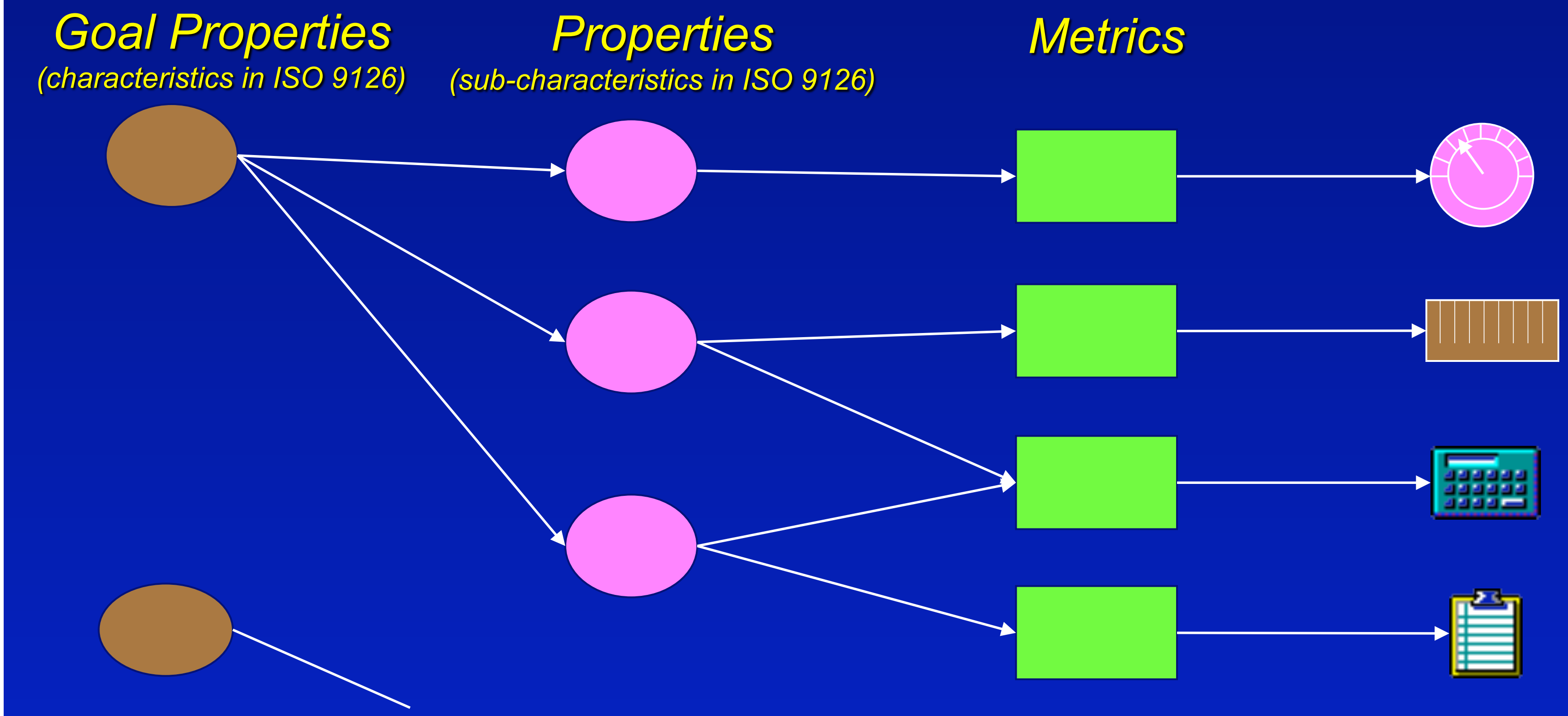
2.3 Past “new challenges”: Independent Verification and Validation (1)

- **For safety and mission critical software,** independent software verification and validation (ISVV) **is required**
  - This requirement is only applicable where the risks associated with the project justify the costs involved
- **ISVV is not merely “independent” testing of the product**
  - The concept of ISVV includes the necessity of setting up **an independent team** of highly qualified staff composed of specialists from all disciplines including software product assurance (“Third Party”)
  - This team, independently of the development team, performs verification and validation activities such as conducting reviews, inspections, testing and auditing
  - The purchaser may also consider a less rigorous level of independence, e.g. an independent team in the same organization

2.3 Past “new challenges”: Independent Verification and Validation (2)

- **ISVV is intended to**
  - ✧ Improve software quality
  - ✧ Reduce development costs
  - ✧ Mitigate project risks
  - ✧ Be complementary to the nominal SW supplier’s V & V tasks
- **ESA Guide for Independent Software Verification and Validation**
  - ✧ Defines the ISVV processes
  - ✧ Disseminates best practices with respect to recommended methods
- **Email INFO-ISVV-GUIDE@esa.int**

3.1 Work in Progress: Software Quality Modeling and Metrication (1)


Goal Properties
(characteristics in ISO 9126)
Properties
(sub-characteristics in ISO 9126)
Metrics

3.1 Work in Progress: Software Quality Modeling and Metrication (2)

□ **Functionality:** The capability of SW to provide functions which meet stated and implied needs when the software is used under specified conditions.

□ **Reliability:** The capability of SW to maintain a specified level of performance when used under specified conditions.

□ **Maintainability:** The capability of the SW product to be modified. Modifications may include corrections, improvements, or adaptation of the software.

□ **Re-usability:** The degree to which a software module or other product can be used in more than one computer program or software system.

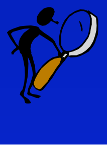
□ **Operability:** The capability of the SW product to enable the user to operate the system or software effectively and correctly.

□ **Documentation Quality:** Those attributes of the software that determine the adequacy of the documentation related to software development, maintenance and operation.

□ **Suitability for Safety:** The capability of the software to allow and contribute to the safe behaviour of the system.

□ **Software Development Effectiveness:** The degree of success/quality of the development process to which the software has been subjected, which provides valuable indications of the product quality.

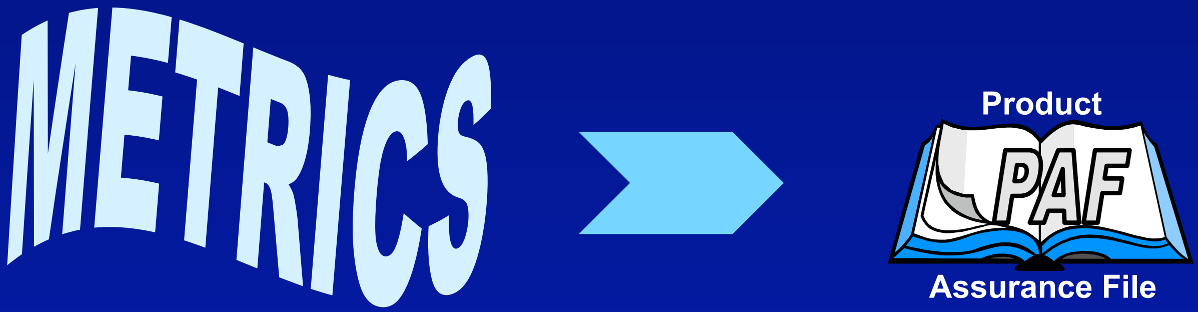
□ **System Engineering Effectiveness:** The level of completeness with which system engineering activities have monitored and controlled the software development process.



3.1 Work in Progress: SW Product Quality Modeling and Metrication (3)

The supplier is in charge of defining the metrics to be used to assess the quality of the product against that specified in terms of these metrics

Supplier



The following basic product metrics shall be used:

- *Size (design and code)*
- *Complexity (design and code)*
- *Fault density and failure intensity*
- *Test coverage*
- *Number of failures*

Q-80B
7.1.8

3.2 Work in Progress: Use of COTS SW & OSS in Critical Functions (1)

COTS SW & OSS dependability: investigate methods to ensure high quality in software systems development using COTS SW & OSS. Detailed objectives:

- Assessment Technical (dependability, integrity) & business (availability, continuity, customer satisfaction) risks related to the use of COTS SW & OSS.
- Definition of software product assurance requirements (process and product related) for the use of COTS SW & OSS. Domain engineering and asset management aspects are relevant.
- Development of guidelines for applying specific software product assurance methods and techniques for the use of COTS SW & OSS. If necessary, adaptation of software product assurance methods and techniques, such as fault and failure analysis and verification practices will be provided;

3.2 Work in Progress: Use of COTS SW & OSS in Critical Functions (2)

Security: investigate security vulnerabilities related to SW systems (COTS SW & OSS related) to assess any impact satellite reliability and security; address the management of the related risks, the integration of security as a discipline in the SW life cycle in the same way as safety and dependability.

Detailed objectives:

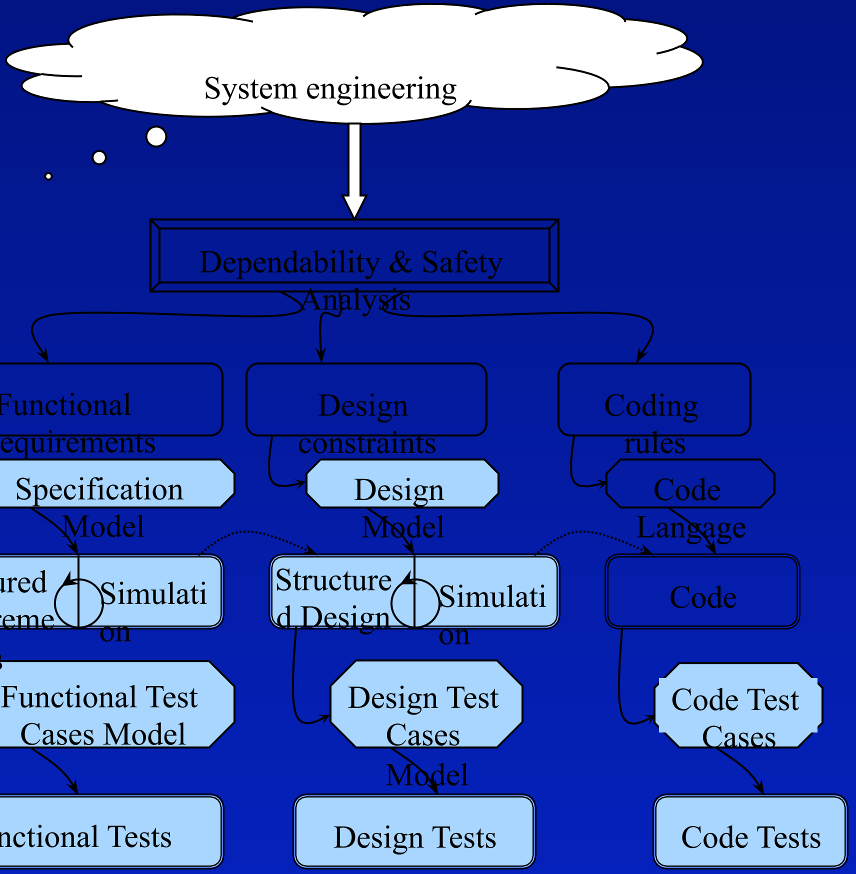
- Perform a risk analysis based on vulnerability analysis and identifying credible software security threats;
- Identify security requirements related to software systems and services and the process used to develop/provide them;
- Identify effective techniques for the implementation in software of security threat prevention, detection and tolerance,
- Introduce security measures in software development, operations, maintenance and support processes;
- Analyse relationships between software safety, dependability, quality and security and trade-off.

3.3 Work in Progress: Model based Validation of RAMS Requirements (1)

- The verification/validation of dependability & safety critical SW components
  - Ensuring that the requirements are complete and consistent.
  - Use of models to facilitate the requirements definition process.
  - Analysis and simulation of the requirements to ensure through metrication the intended completeness and consistency during the early phases.
  - Simulations and analysis on requirements models should support the preparation of high quality validation test suites.

- Introducing validation during a model based requirements definition process should facilitate
  - the verification/validation phase to be reduced in time and effort,
  - effectiveness
  - Preparation of the test suits with the support of the requirements models, for example via simulation or analysis.
  - Demonstration that test suite are suitability thus increasing the confidence that the product meets the quality targets.

3.3 Work in Progress: Model based Validation of RAMS Requirements (2)



Dependability & Safety Requirements

derivation

Requirement Model based

Test cases Model based

Scope of study

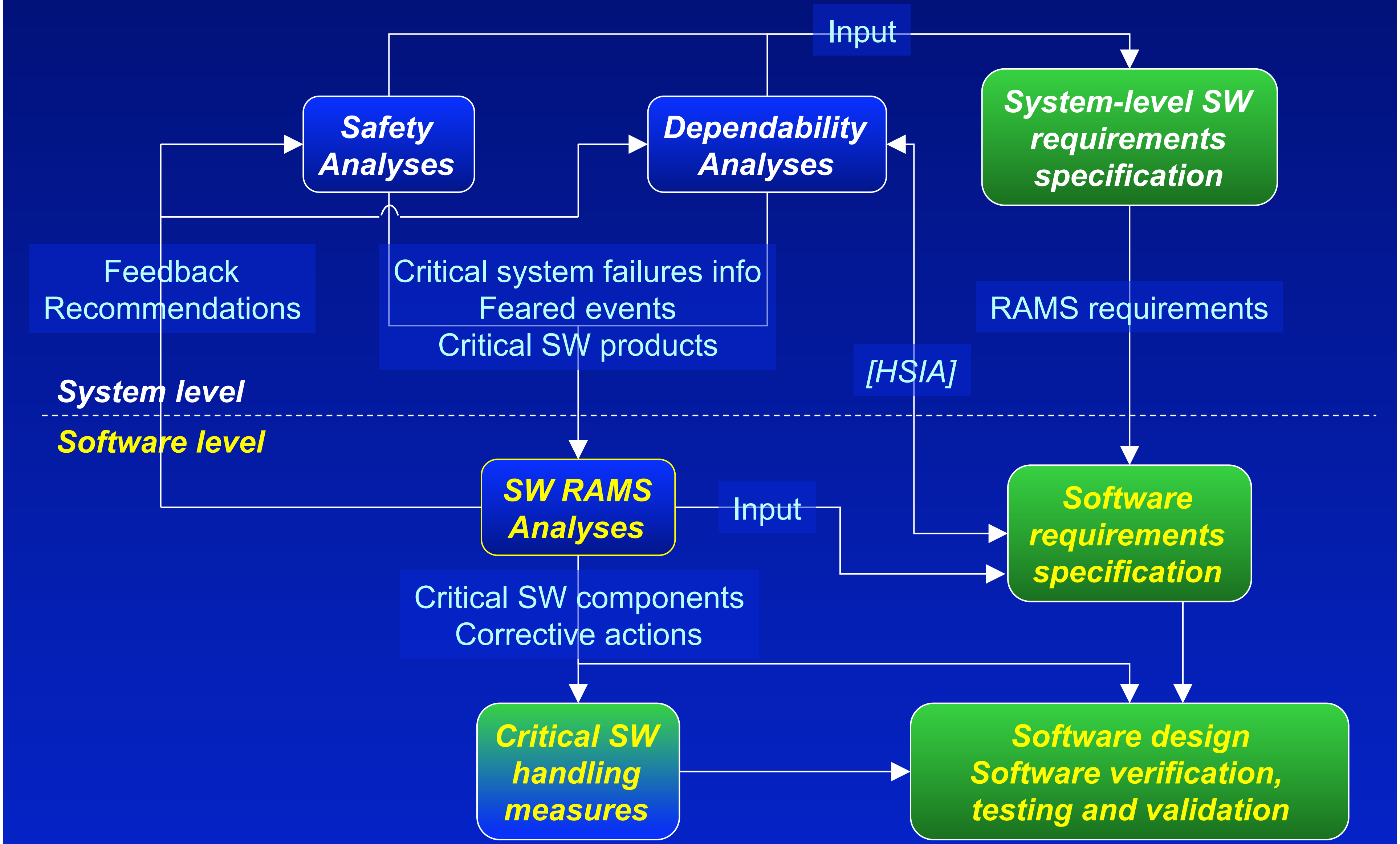
4.1 Little progress: RAMS at Software Level

Why Software RAMS ?

As seen from previous Space Missions, Software plays more and more a central role in space (and non-space) systems. So,

- SW Dependability has to be carefully analysed... Safety is not strictly a Software issue, although the Software behaviour may significantly contribute to the safety of the system …or be a threat to itself!
- Appropriate specification, implementation or verification of SW RAMS requirements are required...(the “root” causes of Software failures).
- The dependability and fail-over robustness of Software have to be assessed and improved.
- SW failures usually end up by having unwanted effects - system failures.
- Design and Organisational constraints have to be identified.

Software RAMS overview (I)



4.1 Little progress: RAMS at Software Level (3)

Application of RAMS methods to a large, software based ground control segment (Scientific Satellite Mission Control System)

Fault Tree Analysis (FTA)

- Starting from list of “Feared Events”, the functional failures (single or combination of failures) leading to the “Feared Events” will be identified
- The functional failures identified shall be classified based on the severity levels of the consequences

Functional Failure Mode Effect Analysis of the MCS (HW&SW)

- Identification of functional failures and the assignment of the severity classification depending on the failure consequences
- All operational phases / modes shall be analysed; the worst potential failure consequences shall be considered when assigning the severity classification

4.1 Little progress: RAMS at Software Level (4)

Application of RAMS methods to a large, software based ground control segment (Scientific Satellite Mission Control System)

Criticality classification of software products

- Mapping the functional failures identified to HW & SW products
- Assigning criticality category to the SW taking into account
  - the mitigation measures implemented in the system (e.g. HW backups, partitioning, monitoring, etc.)
  - the means of intervention and
  - the time available to prevent or mitigate the consequences.

Experimenting with two approaches for SW Criticality Analysis

- Determine the criticality of the each SW architecture component by applying a combination of methods / techniques (e.g. FMECA, FTA, CC/CM analysis) to the software design.
- Update/refinement of the functional MCS FMEA down to the lowest level of definition necessary to analyse the effects of the changes at MCS level.

4.2 Little progress: “Certification” of Software Products (1)

SPEC Evaluation/Certification Process

- ❑ Tailoring of the reference Quality Model based on the criticality classes, space application domains and other project characteristics
  - ✧ Property level
  - ✧ Metric level
- ❑ Selection of the Target Values for metrics
- ❑ Metrics collection and measurements using the selected evaluation method
- ❑ Comparison against the Target Values
- ❑ Graphical visualisation of the results
- ❑ Issue of the certificate (if certification process)
- ❑ Update certification record (if certification process)

Certification could (and should) be prepared in parallel to development, but can only apply to products in a stable state, i.e. at end of development.

4.2 Little progress: “Certification” of Software Products (2)

Why has there not been any success?

- SPEC was “too early”
- Business case result: Space software market too small to be of interest for a certification authority

What is the status today?

- OSS on-board OS (RTEMS) is frequently reused and ESA is interested in having it “certified” for using it in mission critical applications
- ESA catalogue for “certified” products is in preparation; however, this is for “products”, i.e. equipment, components, … also software products -> SPEC is considered to become one element in the certification process

4.3 Little progress: Risk Management using Process Assessment Results (1)

- S4S-R (*S4S-Risk*) is an extension of S4S with a new Risk Dimension
- S4S-R addresses risks arising from inappropriate process management from results of SW process assessments
- Likelihood (Probability)
  - the wider the gap between the target and assessed capabilities, the higher the likelihood of the related risk
- Severity (Impact)
  - the severity of risk's consequences depends upon the capability level in which the gap occurs
- Benefits
  - Focus recommendations on the riskiest processes/practices
  - Minimize the effort of process improvement to achieve acceptable risk level

4.3 Little progress: Risk Model

- Risk vs Process matrix
  - 172 risks (SEI, NTB)
  - 41 processes (S4S)

Risk					
R_3	c_{31}	c_{32}	c_{33}	c_{34}	
R_2	c_{21}	c_{22}	c_{23}	c_{24}	
R_1	c_{11}	c_{12}	c_{13}	c_{14}	
	P_1	P_2	P_3	P_4	Process

- Process Risk Factors (PRF)
  - Weights the impact of a process attribute gap on the overall risk likelihood
  - The lower the capability level at which process attribute gaps occur, the higher the likelihood of the associated risk

Capability Level (CL)	PRF
1	2
2	1.2
3	1
4	0.8
5	0.5

- Risk Dimension Extension included
  - Risk Model
  - Likelihood and Risk Calculation Program
  - GIA Builder

5. Which are not in the List

- Software Reliability Modelling
  - NASA reported conclusion of one of their studies: “Software reliability models have one point in common: they are unreliable”
  - Time and effort demanding
  - Not accepted to support any “qualification” or “safety” argument
  - Not suitable for ultra high reliability or safety applications (for low criticality applications nobody would make the resources available anyhow)
- N-version programming
  - Not even specified in any of our software standards as an “option”
  - Explicitly discouraged in one of our main projects (Galileo)
  - Nobody convinced that this would result in more reliable software
  - Required resources are better invested in extensive V & V activities
  - However, it is used in Airbus (please correct me if I am wrong)

Conclusion and Outlook

ESA Software Projects rely on

- Mature Processes, proven methods and standards (no experiments)
- Extensive Verification, Validation and Testing (including “third party”)
- Software Reuse for “standard” functions and components
- Simple quality modelling and metrication (e.g. test coverage, trends in problem reports)
- Extensive documentation

More and more used (or planned) are:

- COTS in critical functions
- Advanced quality modelling and metrication (process and product)
- RAMS on software level
- Certified product (to be managed in a space product catalogue)

HARMONIZATION OF R & D BETWEEN ESA, NATIONAL SPACE AGENCIES & SPACE INDUSTRY IS FORESEEN IN 2009