

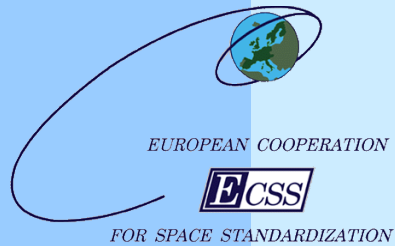
# **ECSS-Q-ST-80C**

## **Space product assurance**

---

## **Software product assurance**

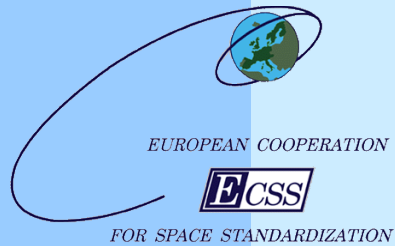
### ***Training Course***



### **COPYRIGHT NOTICE:**

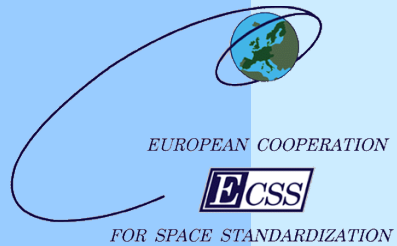
**By using the ECSS Training material, developed by ESA, you agree to the following conditions:**

- 1. The training shall take place at your premises and shall be addressed to your staff (internal participants);**
- 2. In case of a training to be given to external participants, the prior ESA written authorisation shall be requested;**
- 3. The ESA Copyright shall always be mentioned on all Training Material used for the purpose of the training and participants shall acknowledge the ESA ownership on such a Copyright;**
- 4. The Training material shall not be used to generate any revenues (i.e. the training and Training Material shall be "free of charge" excl. any expenses for the training organisation);**
- 5. Only non-editable PDF files of the Training Material can be distributed to the participants (nor power point presentations);**
- 6. Any deficiency identified in the Training Material shall be reported to the ECSS secretariat;**
- 7. If the Training Material is modified or translated, the ESA Copyright on such edited Training Material shall be clearly mentioned. A copy of the edited Training Material shall be delivered to ESA for information.**
- 8. You shall always hold harmless, indemnify and keep ESA indemnified against any and all costs, damages and expenses incurred by ESA or for which ESA may become liable, with respect to any claim by third parties related to the use of the Training Material.**



# ECSS-Q-ST-80C

- **Origins and Status**
- **Key Concepts**
- **Organization**
  - Software Product Program Implementation
  - Software Process Assurance
  - Software Product Quality Assurance
- **Handbooks**



# Origins and Status of the ECSS-Q-ST-80 Standard

# Software Product Assurance

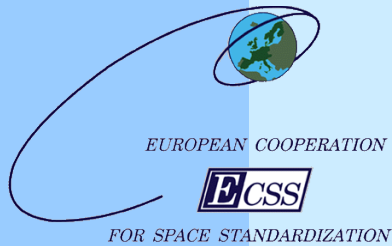
## Software Product Assurance (SPA)

3.2.23


- *the totality of activities, standards, controls and procedures in the lifetime of a software product which establish confidence that the delivered software product, or software affecting the quality of the delivered product, conforms to customer requirements*


### Objectives:

provide adequate confidence to the customer and to the suppliers that developed or reused software satisfies the requirements throughout the system lifetime




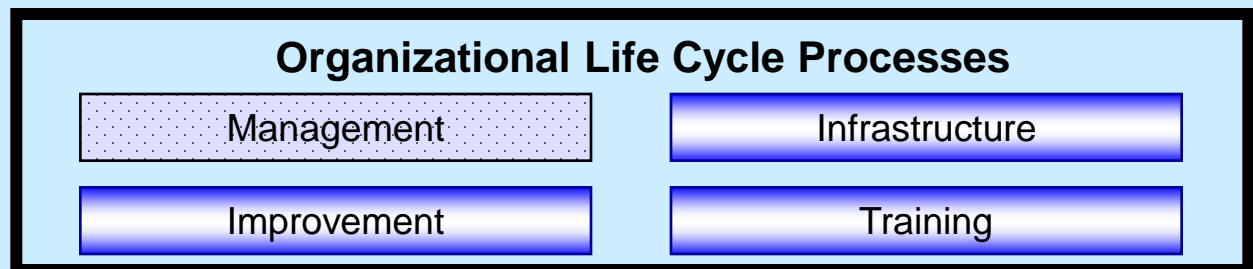
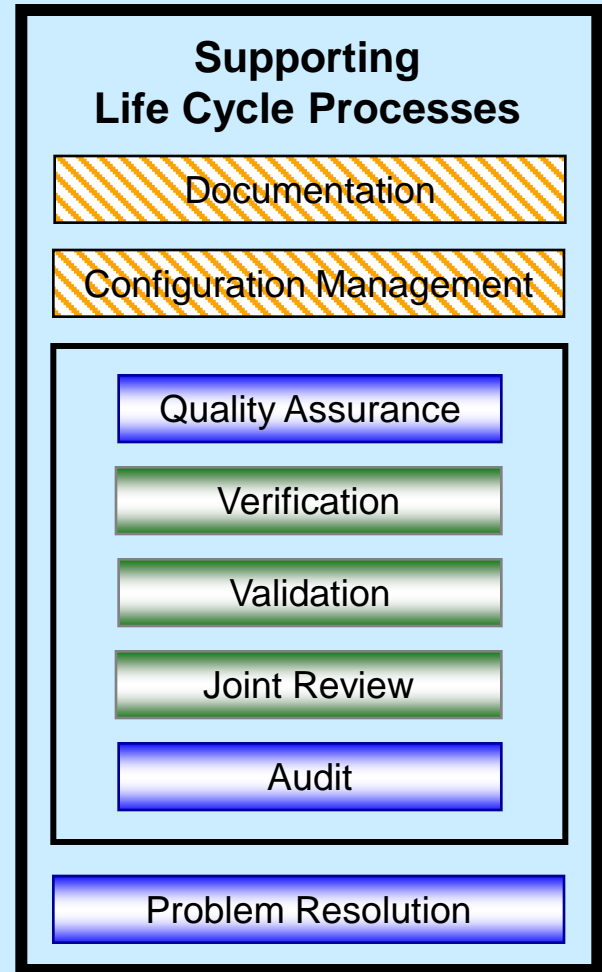
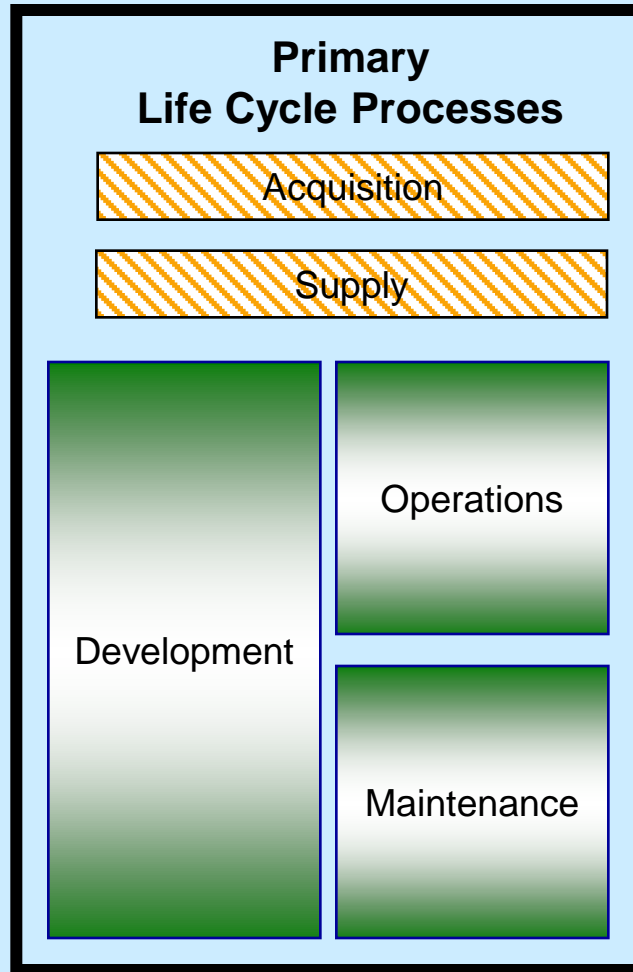
# Based on ISO/IEC 12207

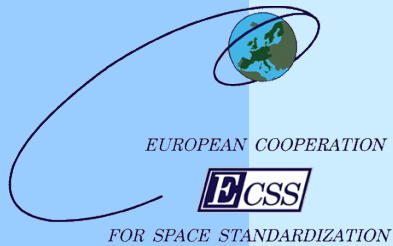
Other ECSS 

E-40B 

Q-80B 

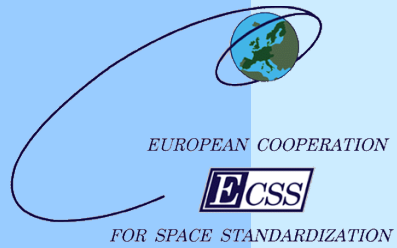
Details for SPA and/or SWE 





# Evolution of ECSS-Q-80

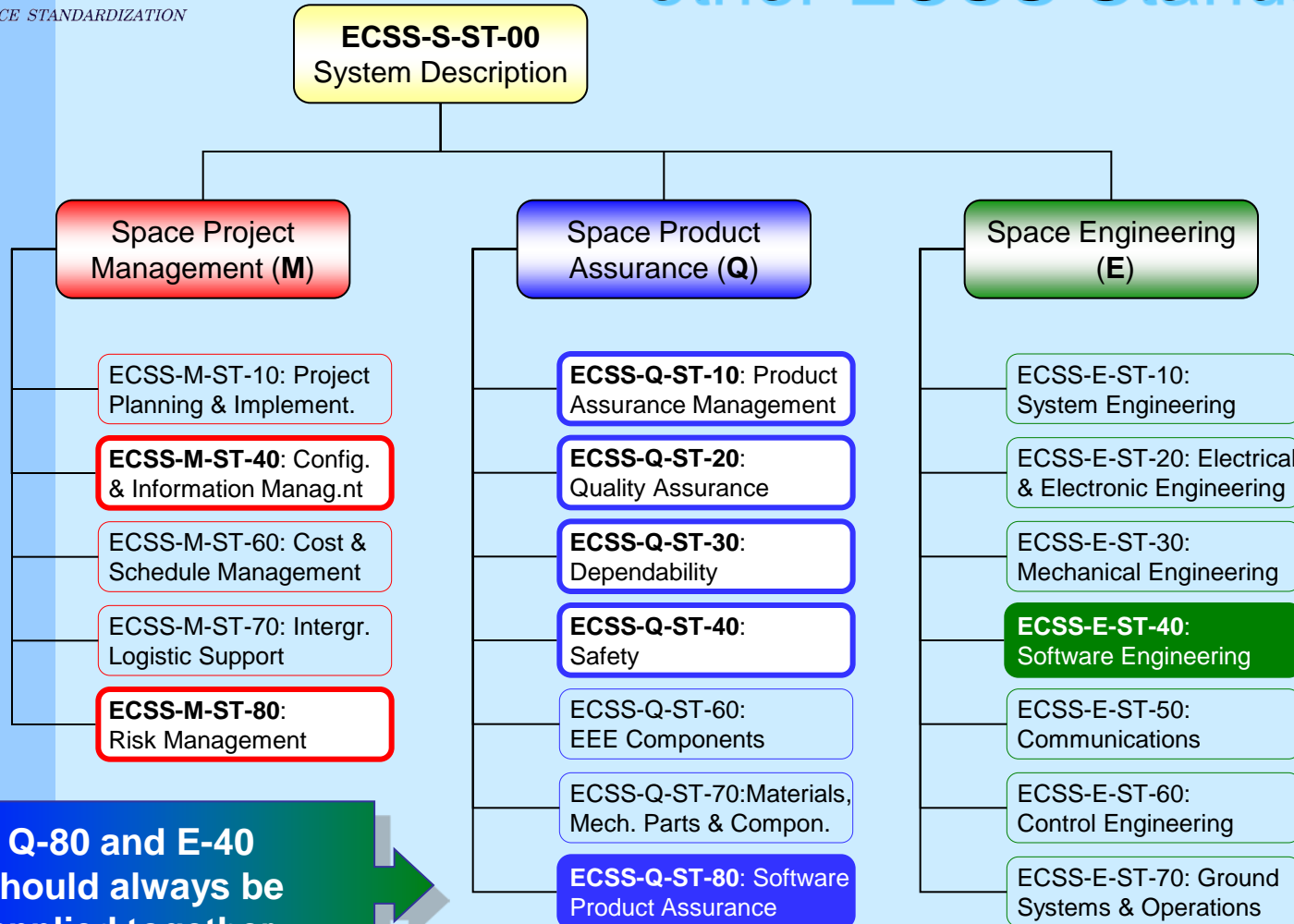
- **ECSS-Q-80A**            **April 1996**  
***First Issue***
  - *Contains several engineering requirements*
  
- **ECSS-Q-80B**            **October 2003**  
***Second Issue***
  - *Updated and harmonized with ECSS-E-40B*
  
- **ECSS-Q-ST-80C**    **March 2009**  
***Current Issue***
  - *Updated and streamlined in accordance with ECSS Task Force 2 recommendations*



# Key Concepts in the ECSS-Q-ST-80 Standard



# The relation to the other ECSS Standards



# The customer-supplier relationship

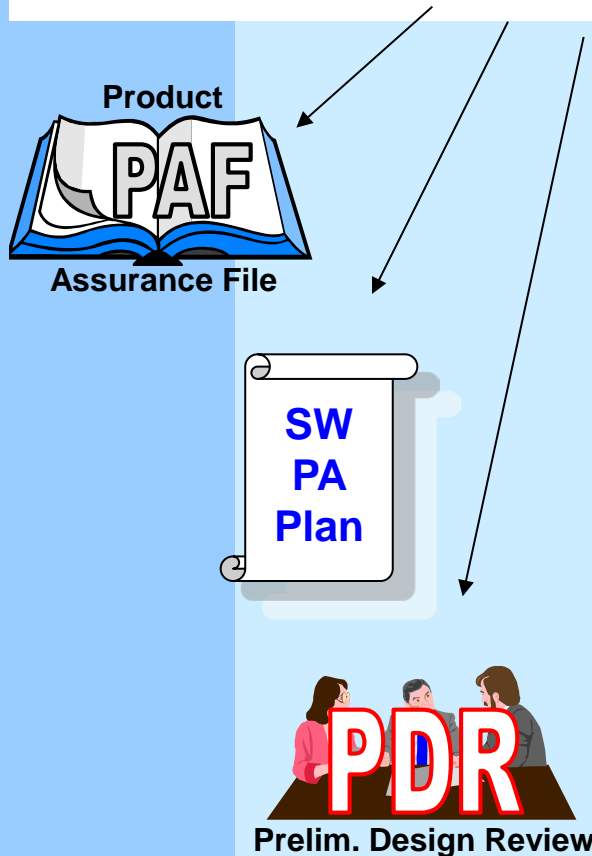
- Described in **ECSS-S-ST-00 (6.1)**
- Applied recursively (customer-supplier **chain**)
  - Top level customer (typically an Agency, e.g. ESA)
  - Intermediate levels: *both customer and supplier*
  - Lowest level: supplier only



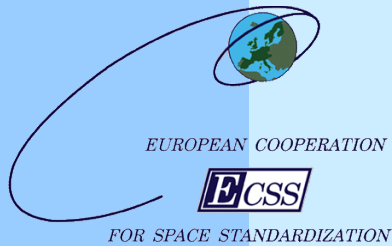
- For software development, the customer is often **internal**
  - “System level”

# Requirements & Expected Output

*EXPECTED OUTPUT: Software product assurance plan [PAF, SPAP; PDR].*

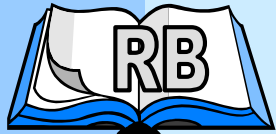


- Each Q-80B requirement is identified by a **hierarchical number**
- For each requirement, the associated **Expected Output** is specified
- The Expected Output includes:
  - the **destination file**
  - the **DRD** of the document containing the output
  - the **reviews** at which the output must be provided

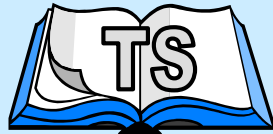


# Software documentation

**Annex A**



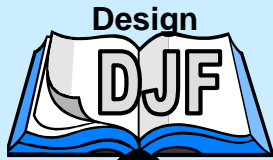
Requirements Baseline



Technical Specification



Design Definition File



Justification File



Management File



Operational Documentation

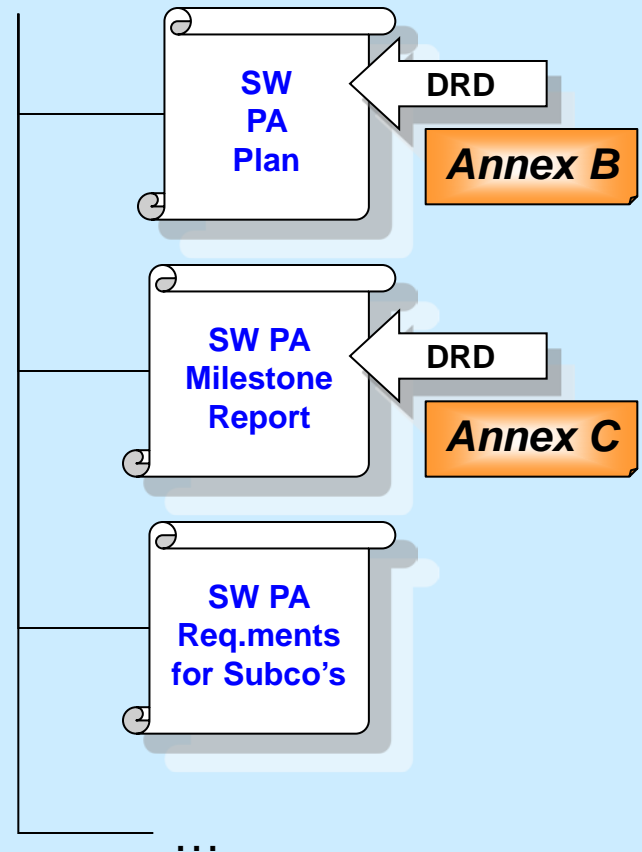


Maintenance File

**Files (a.k.a. folders)  
are shared with  
ECSS-E-ST-40**



Product Assurance File



# The concept of tailoring

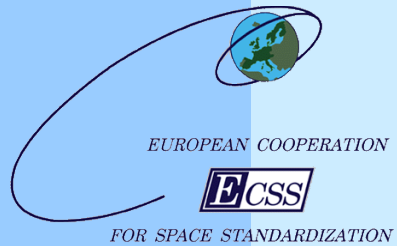
- ECSS Standards should not be made applicable “as is” ⇒ **tailoring required**
- Tailoring is a **customer’s responsibility**
- ECSS-Q-ST-80C contains a tailoring **matrix** based on software criticality

## Annex D

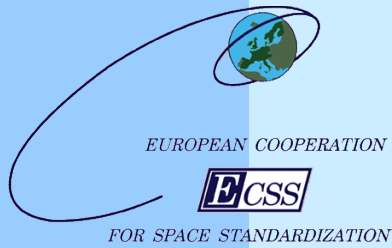
Requirements applicable to ECSS-E-ST-40 processes that are tailored out are automatically not applicable



SW Product  
**Q-80C**  
Assurance



# Organization of the ECSS-Q-ST-80 Standard

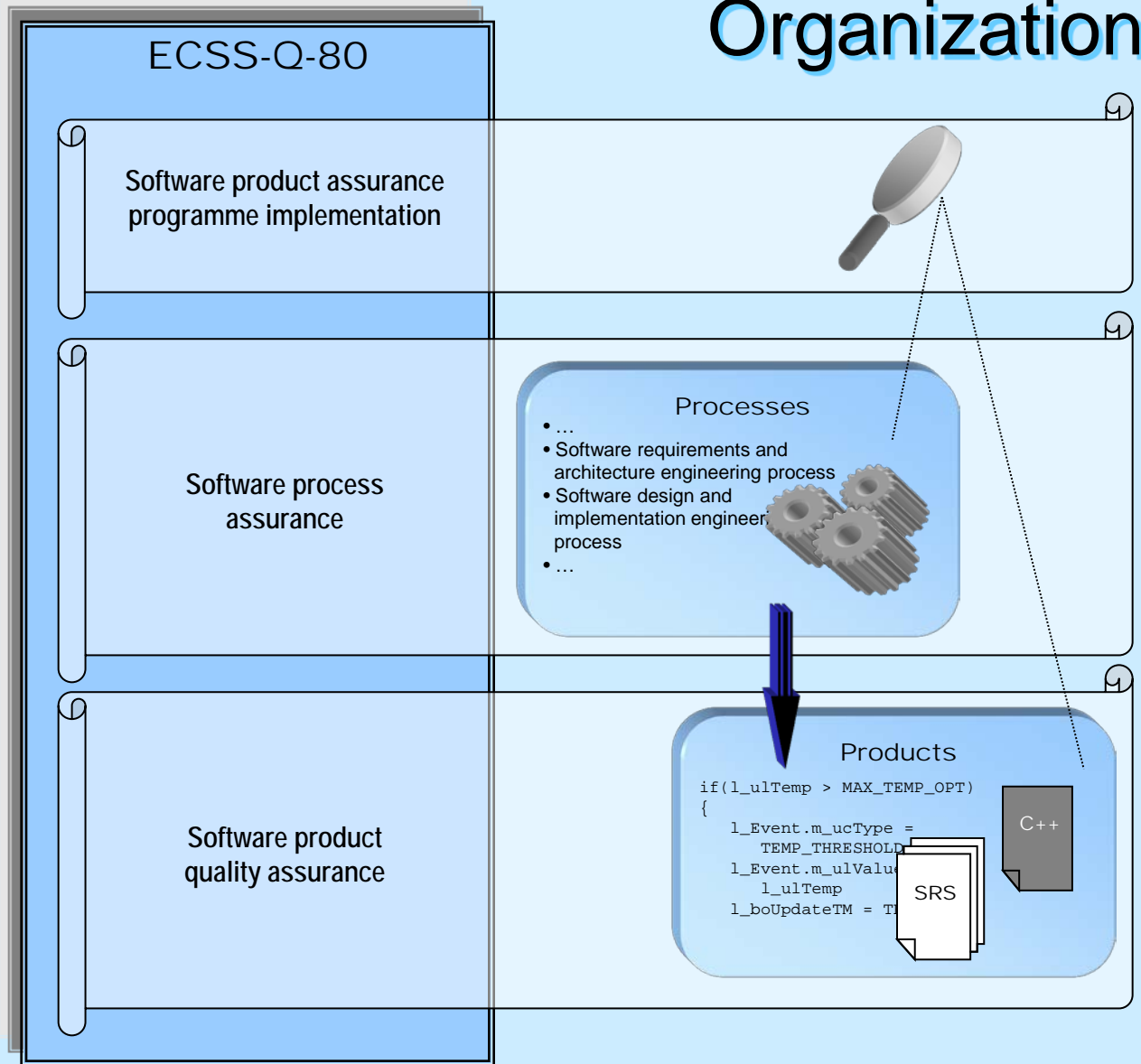


# Organization

**Clause 5**

**Clause 6**

**Clause 7**



# Structure

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

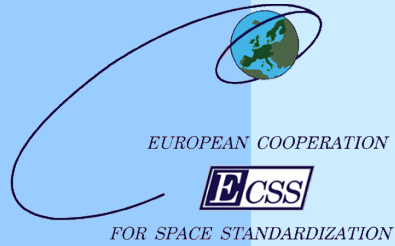
## Software process assurance

- 6.1 Software development life cycle
- 6.2 Requirements applicable to all software engineering processes
- 6.3 Requirements applicable to individual software engineering processes or activities

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

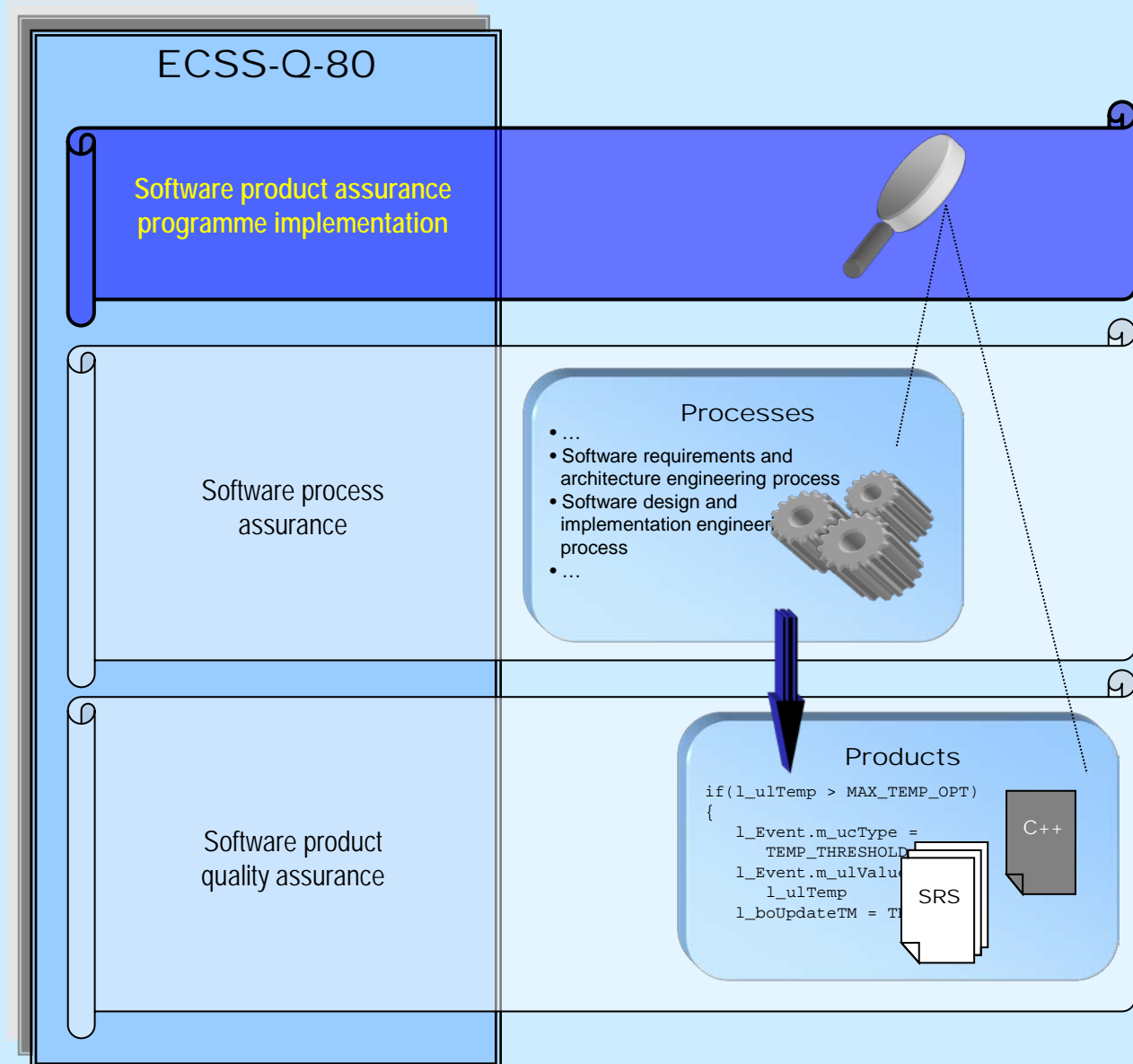




# Software Product Assurance Process Implementation

## Clause 5

# Setting up and running a SW PA programme



# Organization and Responsibility

## Software product assurance programme implementation

- |   |                                      |
|---|--------------------------------------|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control   |
| 5.2 Software product assurance programme management | 5.5 Procurement                      |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment |
|   | 5.7 Assessment and improvement       |

- 5.1.1 Organization
- 5.1.2 Responsibility and authority
- 5.1.3 Resources
- 5.1.4 Software product assurance manager/engineer
- 5.1.5 Training

## Software product assurance

- 6.1 Software product assurance requirements
- 6.2 Requirements applicable to individual software engineering processes or activities
- 6.3 Requirements applicable to individual software engineering processes or activities

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

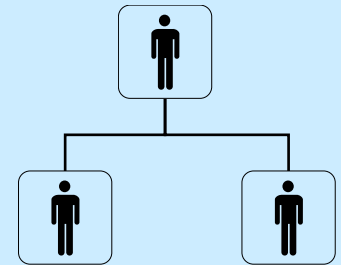
# Organization and Responsibility

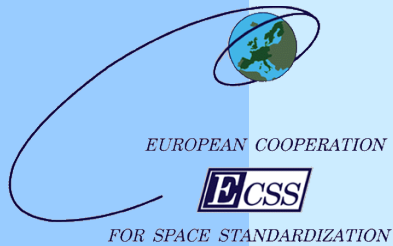
## 5.1.1

- An organizational structure for **software development** must be defined
- For personnel whose work affects **quality**:

## 5.1.2

- Roles
- Responsibilities
- Authority
- Interrelations
- Interfaces to other organizations (internal and external)
- Delegation of PA tasks to lower-level suppliers



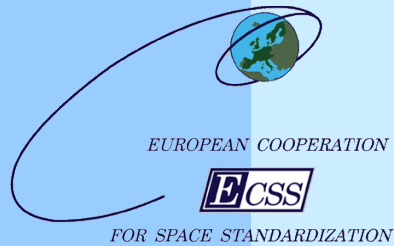


# Resources

## 5.1.3

- The supplier must identify resource requirements for the **software product assurance** function
- Resources must be allocated and made available for the **SPA** tasks
- The **independence** of personnel performing reviews and audits must be ensured
  - Not involved in the processes being reviewed/audited

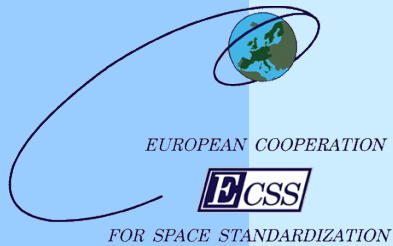
Reflects ECSS-Q-ST-10C, subclause 5.1.1.3



# Software Product Assurance Manager/Engineer

## 5.1.4

- The supplier must identify the personnel in charge of the **software product assurance** tasks
- The traditional title of **software product assurance manager** may not be applicable ⇒ engineer
- **He/she:**
  - **Reports** to Project Manager (via PA manager, if any)
  - Has got **authority and independence** to carry out his/her tasks
  - Has got **unimpeded access** to higher management



# Training

## 5.1.5

- **The supplier must timely **plan** to ensure that:**
  - the **resources and skills** needed for all the staff are acquired or developed
  - the right composition and categories of **appropriately trained personnel** are available
- **The **training subjects** must be determined by the specific tools, techniques, methodologies and computer resources to be used**
- **A **training plan** must be produced and **training records** must be maintained**

# Software product assurance programme management

## Software product assurance programme implementation

- |  |  |
|--|--|
| 5.1 Organization and responsibility                        | 5.4 Supplier selection and control     |
| <b>5.2 Software product assurance programme management</b> | 5.5 Procurement                        |
| 5.3 Risk management and critical item control              | 5.6 Tools and supporting environment   |
|  | 5.7 Assessment and improvement process |

- 5.2.1 Software product assurance planning and control
- 5.2.2 Software product assurance reporting
- 5.2.3 Audits
- 5.2.4 Alerts
- 5.2.5 Software problems
- 5.2.6 Nonconformances
- 5.2.7 Quality requirements and quality models

## Software product assurance

- 6.1 Software product assurance
- 6.2 Requirements
- 6.3 Requirements or activities

## Software product assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

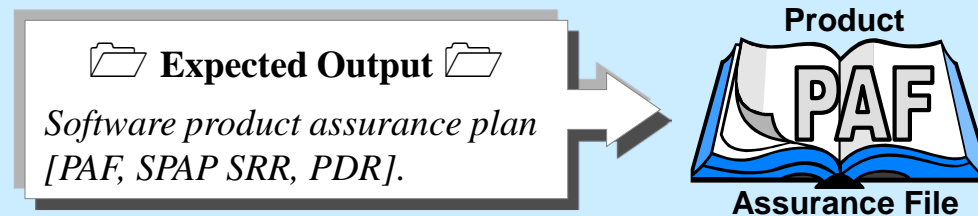


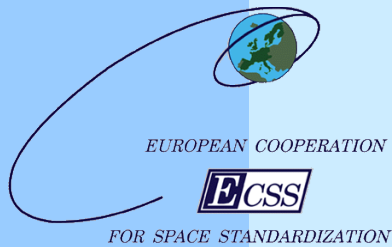
# Software product assurance planning and control

## 5.2.1

- A **Software Product Assurance Plan** must be produced and delivered for customer approval
  - May be part of the **overall** project PA plan
- The **SPAP** must be up-to-date at each milestone
- The **SPAP** must contain a **compliance matrix** with respect to the applicable requirements
  - For each requirement, **reference** to the implementation of that requirement
- A **DRD** for the **SPAP** is provided in Annex B

## Annex B





# Software product assurance reporting

5.2.2

Annex C

- Regular **software product assurance reporting** to be provided as part of the overall project reporting
- Specific **software product assurance milestone reporting** to be provided at milestone reviews
- **Main reporting subjects:**
  - Assessment of product and process quality
  - Verifications undertaken
  - Problems detected and resolved



# Audits and Alerts

## 5.2.3

- **Audits** to be performed in accordance with **ECSS-Q-ST-10, subclause 5.2.3**
  - Internal and external (on suppliers)
  - To verify the implementation of the (software) product assurance programme

## 5.2.4

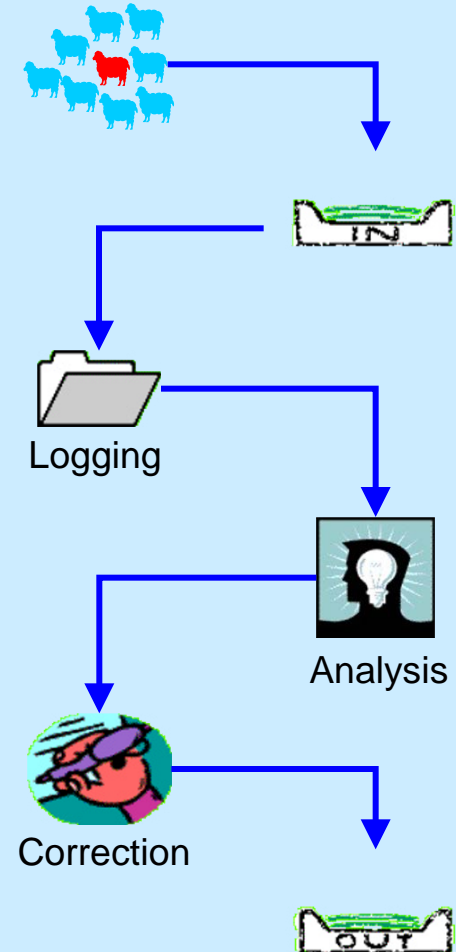
- **Alerts** to be treated in accordance with **ECSS-Q-ST-10, subclause 5.2.9**
  - The supplier to participate in the alert system organized by the customer or other sources
  - **ESA** maintains an alert system



# Software problems

## 5.2.5

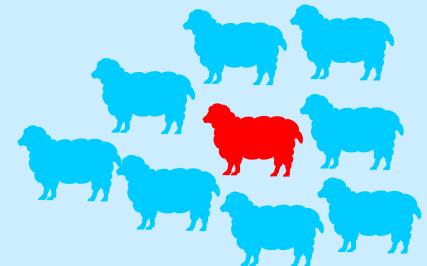
- The supplier must define and implement **procedures** for software problems logging, analysis and correction
- Detailed content of **software problem report (SPR)** specified
- Interface with **nonconformance** system to be defined
- **Verify** correct implementation of SPR procedures



# Nonconformances

## 5.2.5

- The supplier must define and implement a **nonconformance control system** in accordance with **ECSS-Q-ST-10-09**
  - Identification, classification, segregation, reporting, review, analysis, disposition of NCRs
  - Corrective and preventive actions
- **SW product assurance** and **SW engineering** must be represented in nonconformance review boards
- The supplier to specify **when** NCR management for software starts



# Quality requirements and quality models (1/2)

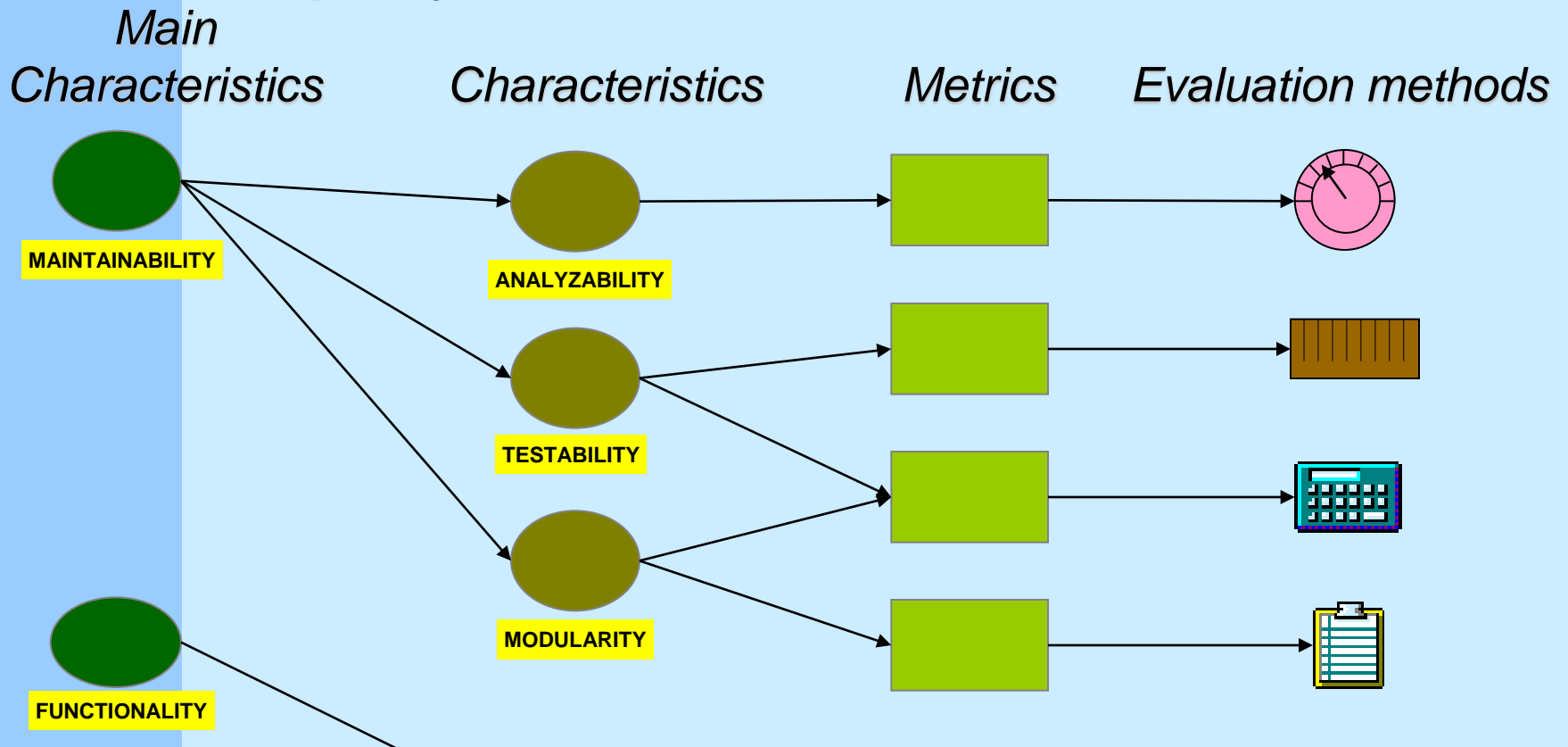
- **Quality models must be used to specify quality requirements**
- **Quality characteristics:**
  - functionality
  - reliability
  - maintainability
  - reusability
  - suitability for safety
  - security
  - usability
  - efficiency
  - portability
  - software development effectiveness

5.2.7

Quality models are defined e.g. in  
ISO 9126  
ECSS-Q-HB-80-04  
(to be published)

# Quality requirements and quality models (2/2)

- Quality models allow to specify and measure quality



# Risk management and critical item control

## Software product assurance programme implementation

- |  |  |
|--|--|
| 5.1 Organization and responsibility                  | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management  | 5.5 Procurement                        |
| <b>5.3 Risk management and critical item control</b> | 5.6 Tools and supporting environment   |
|  | 5.7 Assessment and improvement process |

## Software process assurance

- |  |  |
|--|--|
| 6.1 Software process requirements  | <b>5.3.1 Risk management</b><br><b>5.3.2 Critical item control</b> |
| 6.2 Requirements applicable to individual software engineering processes or activities |  |
| 6.3 Requirements applicable to individual software engineering processes or activities |  |

## Software product quality assurance

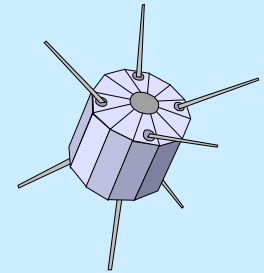
- |  |
|--|
| 7.1 Product quality objectives and metrication                   |
| 7.2 Product quality requirements                                 |
| 7.3 Software intended for reuse                                  |
| 7.4 Standard ground hardware and services for operational system |
| 7.5 Firmware   |



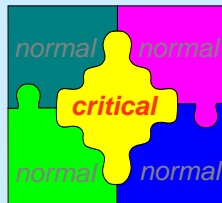
# Risk management and critical item control

## 5.3.1

- The supplier must perform **risk management** in accordance with ECSS-M-ST-80
- Software risk management contributes to the **overall project risk policy**



## 5.3.2



- The supplier must perform **critical item control** in accordance with ECSS-Q-ST-10-04
- The **characteristics** which make SW candidate for C.I.L. must be identified

# Risk management and critical item control

## Software product assurance programme implementation

- |   |                                      |
|---|--------------------------------------|
| 5.1 Organization and responsibilities               | 5.4 Supplier selection and control   |
| 5.2 Software product assurance programme management | 5.5 Procurement                      |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment |
|   | 5.7 Assessment and improvement       |

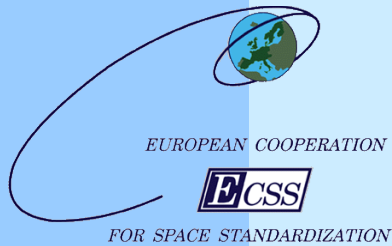
- 5.4.1 Supplier selection
- 5.4.2 Supplier requirements
- 5.4.3 Supplier monitoring
- 5.4.4. Criticality classification

## Software product quality assurance

- 6.1 Requirements applicable to all software engineering processes
- 6.2 Requirements applicable to all software engineering processes
- 6.3 Requirements applicable to individual software engineering processes or activities

## Software product quality assurance

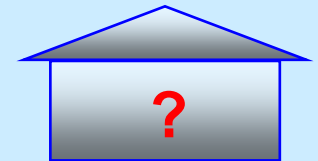
- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

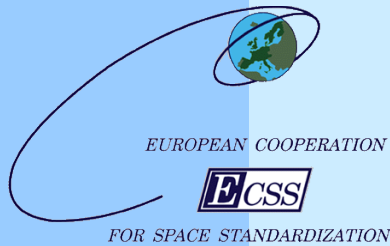


# Supplier selection

## 5.4.1

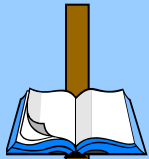
- The supplier must **select lower level suppliers** following the process specified in ECSS-Q-ST-20C, subclause 5.4.1
  - QA function must be involved in selection
  - Criteria for selection (e.g. pre-award audit)
  - Records of suppliers to be maintained
- For suppliers of **COTS software**, the reuse file must be made available and used in the selection process
  - Selection of and negotiations with suppliers of COTS software based on knowledge of the actual status of the software being procured





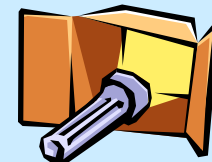
# Supplier requirements and monitoring

## 5.4.3

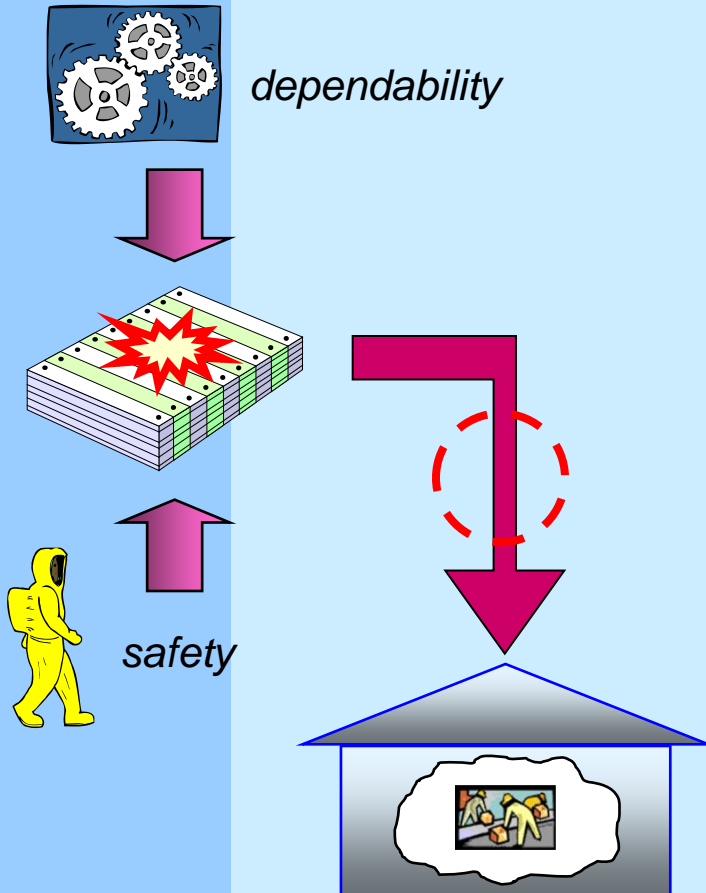


## 5.4.2

- The supplier must establish **software product assurance requirements** for lower level suppliers
  - To be approved by the customer
- The supplier must **monitor** the compliance of his suppliers with the **SW PA** requirements
  - Verify definition and implementation of software development processes
  - Suppliers' product assurance plan to be provided to customer for approval
  - Continuous verification of processes and products



## 5.4.4



# Criticality classification

- The supplier must provide the lower level suppliers with the results of the **RAMS analyses** performed:
  - at higher level
  - at his level
- The lower level suppliers get:
  - **criticality classification** of the software being procured
  - **information on failures** that led to that criticality classification

Linked to SW RAMS (6.2.2)

# Procurement

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | <b>5.5 Procurement</b>                 |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

### 5.5.1 Procurement documents

### 5.5.2 Review of procured software component list

### 5.5.3 Procurement details

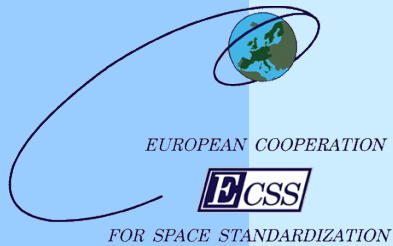
### 5.5.4 Identification

### 5.5.5 Inspection

### 5.5.6 Exportability

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware



# Procurement

## 5.5

- For **procured** (non-developed) software
- **ECSS-Q-ST-20C, subclause 5.4.2, applies**
  - Clear, complete and traceable **requirements**
  - **QA requirements** to be included
  - **QA function** to review the procurement documentation
- **Procured software details to be provided to the customer**
- **Procured software must be inspected and put under configuration control**
- **Mind exportability**



# Tools and supporting environment

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

### 5.6.1 Methods and tools

### 5.6.2 Development environment selection

- 6.1
- 6.2 Requirements applicable to all software engineering processes
- 6.3 Requirements applicable to individual software engineering processes or activities

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware



**5.6.1**

- The supplier must identify in the SPAP all used **methods** and **tools**
- **Justify** the choice of those methods and tools in the SPAMR
  - The team knows how to use them
  - They are appropriate and available for this project
- **Verify** the correct use of methods and tools

# Methods and tools



# Software development environment

## 5.6.2

- **Development environment to be chosen based on specific criteria**
  - availability
  - performance
  - maintenance
  - ...
- **Justification of suitability** for the current project to be provided to the customer
- The development environment must be **available** to the team at the start of each phase



# Assessment and improvement process

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

So

- 5.7.1 Process assessment
- 6. 5.7.2 Assessment process
- 6. 5.7.3 Process improvement
- 6.

or activities

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

# Process assessment

## 5.7.1

- The supplier must **monitor and control** the effectiveness of the processes used during the development of the software
- Not necessarily performed at **project level**
  - The process assessment and improvement performed at **organization level** can be used to provide evidence of compliance for the project
- The supplier must provide evidence that a **process assessment and improvement process is in place** in its organization
  - **Records** to be made available



# Assessment process

## 5.7.2

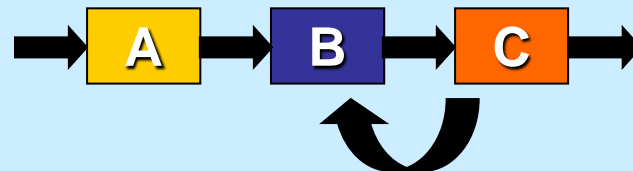
- The **process assessment model and method** used to perform software process assessment must be documented
- Models and actual assessments must comply with **SPICE (ISO/IEC 15504)**
  - But...**CMMI** model + **SCAMPI** A methods are OK
  - **ECSS-Q-HB-80-02** contains a method and a model which are conformant to ISO 15504
- **Assessments must be performed by skilled personnel**
  - ISO 15504 ⇒ **competent assessor**
  - CMMI ⇒ SEI authorized **lead appraiser**

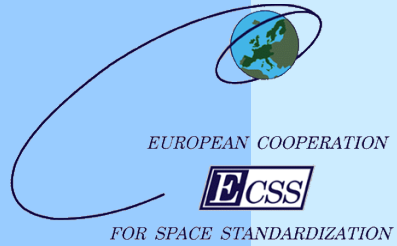


# Process improvement

## 5.7.3

- Use the **results** of process assessment to:
  - improve the processes
  - recommend **changes**
  - determine technology **needs**
- **Process improvement process to be documented**
  - ECSS-Q-HB-80-02 and CMMI Organization Process Focus provide guidance
- **Evidence of improvement to be documented**

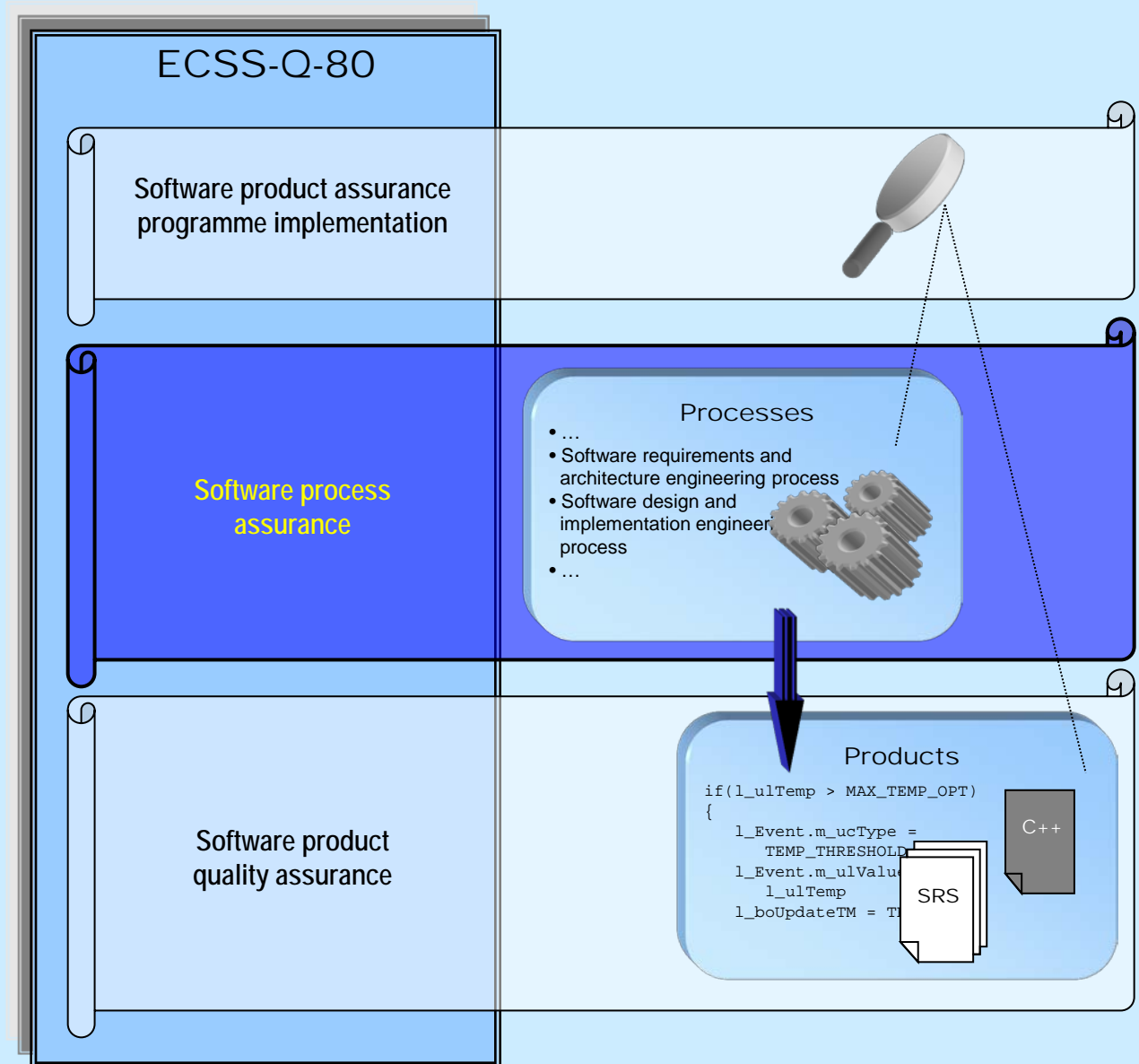




# Software Process Assurance

## Clause 6

# Ensuring the quality of the process





# Software development life cycle

## Software product assurance programme implementation

- 5.1 Organization and responsibilities
- 5.2 Software product assurance programme management
- 5.3 Risk management and configuration item control

- 6.1.1 Life cycle definition
- 6.1.2 Process quality objectives
- 6.1.3 Life cycle definition review
- 6.1.4 Life cycle resources
- 6.1.5 Software validation process schedule

## Software process assurance

- 6.1 Software development life cycle**
- 6.2 Requirements applicable to all software engineering processes
- 6.3 Requirements applicable to individual software engineering processes or activities

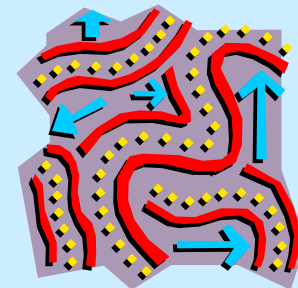
## Software product quality assurance

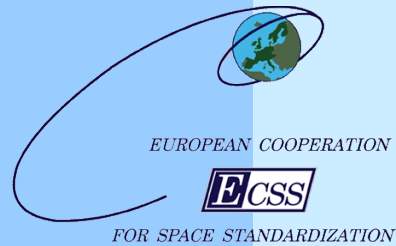
- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

# Life cycle definition

## 6.1.1

- The **software development life cycle** must be defined in the software development plan and referenced in the SPAP
  - If not defined in the **development plan**, the life cycle must be defined in the **SPAP**
- **Life cycle characteristics** to be provided
  - phases and milestones
  - input, output and state of completion at end of phase
  - dependencies
  - responsibilities
  - role of customer at reviews





# Life cycle: quality objectives, review, resources and validation

## 6.1.2

- Use **quality objectives** when defining life cycle

## 6.1.3

- Software life cycle must be **reviewed**

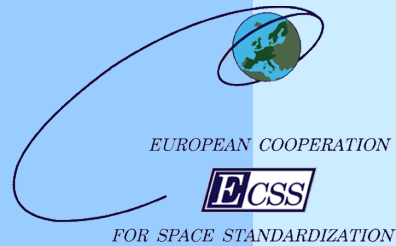
## 6.1.4

- against **contractual requirements**
- for **suitability** and **availability of resources**

## 6.1.5

- A **milestone** must be scheduled immediately before the starting of software validation
  - check if **software status** is compatible
  - check if **resources, documents** and **equipments** are OK to start validation
  - software **Test Readiness Review**





# Requirements applicable to all software engineering processes

## Software product assurance

- 5.1 Organization and responsibilities
- 5.2 Software product assurance programme management
- 5.3 Risk management and configuration item control

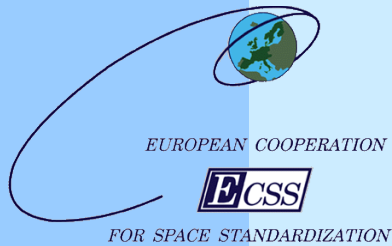
## Software process assurance

- 6.1 Software development life cycle
- 6.2 Requirements applicable to all software engineering processes**
- 6.3 Requirements applicable to individual software engineering processes or activities

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

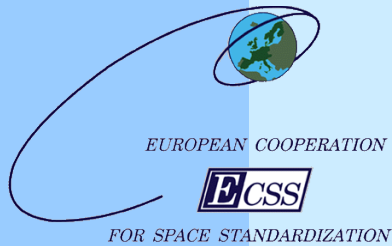
- 6.2.1 Documentation of processes
- 6.2.2 Software dependability and safety
- 6.2.3 Handling of critical software
- 6.2.4 Software configuration management
- 6.2.5 Process metrics
- 6.2.6 Verification
- 6.2.7 Reuse of existing software
- 6.2.8 Automatic code generation



# Documentation of processes (1/2)

## 6.2.1

- **The supplier must generate **plans** (either software specific or at project level) to cover:**
  - development;
  - specification, design and customer documents to be produced;
  - configuration and documentation management;
  - verification, testing and validation activities;
  - maintenance.
  
- **The **SPAP** identifies all plans to be produced and used, the relationship between them and the time-scales for their preparation and update.**



## Documentation of processes (2/2)

### 6.2.1

- The supplier must generate or identify **procedures and standards** to be used for all type of software in the project
- Plans must be reviewed against **contractual requirements**
- Procedures and standards must be reviewed against **plans** and contractual requirements
- Plans, procedures and standards must be **finalized** before the start of the activities
- Plans must be **updated** at each milestone to reflect development changes

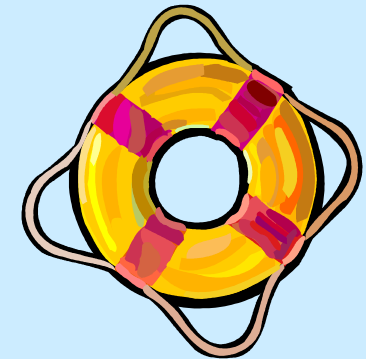
# Software dependability and safety (1/7)

## 6.2.2

### ■ Software **RAMS**

- Reliability
- Availability
- Maintainability
- Safety

} Dependability



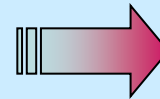
- **Software **RAMS activities** start at system level and continue at software level, with feedback**
- **Main objectives:**
  - identify the **critical software**
  - define and implement **measures** to handle the critical software

# Software dependability and safety (2/7)

## 6.2.2.1

### ■ At **system level**:

- Dependability analyses  
(**ECSS-Q-ST-30**)
- Safety analyses  
(**ECSS-Q-ST-40**)



Criticality classification  
of system functions  
and *products*



based on severity classification of system  
failures consequences



**Failure  
consequences  
severity  
categories**

# Software dependability and safety (3/7)

<b>NAME</b>	<b>LEVEL</b>	<b>DEPENDABILITY (ECSS-Q-30)</b>	<b>SAFETY (ECSS-Q-40)</b>
<b>CATASTROPHIC</b>	1		<ul style="list-style-type: none"> <li>• LOSS OF LIFE, LIFE-THREATENING OR PERMANENTLY DISABLING INJURY OR OCCUPATIONAL ILLNESS.</li> <li>• LOSS OF AN INTERFACING MANNED FLIGHT SYSTEM</li> <li>• SEVERE DETRIMENTAL ENVIRONMENTAL EFFECTS.</li> <li>• LOSS OF LAUNCH SITE FACILITIES.</li> <li>• LOSS OF SYSTEM</li> </ul>
<b>CRITICAL</b>	2	COMPLETE LOSS OF MISSION	<ul style="list-style-type: none"> <li>• TEMPORARILY DISABLING BUT NOT LIFE-THREATENING INJURY, OR TEMPORARY OCCUPATIONAL ILLNESS .</li> <li>• MAJOR DETRIMENTAL ENVIRONMENTAL EFFECTS.</li> <li>• MAJOR DAMAGE TO PUBLIC OR PRIVATE PROPERTIES.</li> <li>• MAJOR DAMAGE TO INTERFACING FLIGHT SYSTEMS,</li> <li>• MAJOR DAMAGE TO GROUND FACILITIES.</li> </ul>
<b>MAJOR</b>	3	MAJOR MISSION DEGRADATION	
<b>MINOR OR NEGLIGIBLE</b>	4	MINOR MISSION DEGRADATION OR ANY OTHER EFFECT.	

# Software dependability and safety (4/7)

critical software

**Software that if not executed, or if not correctly executed, or whose anomalous behaviour could cause or contribute to a system failure resulting in:**

consequences

Defined in  
Q-30 & Q-40

A

**Catastrophic**

B

**Critical**

C

**Major**

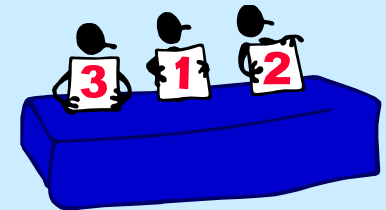
D

**Minor or negligible**

# Software dependability and safety (5/7)

## 6.2.2.2

- The supplier must perform a **software dependability and safety analysis** to determine the criticality category of software *components*



## 6.2.2.3

- Analysis to be performed **at technical specification and design level**, e.g.:
  - SFMECA
  - SFTA
  - SCCA



## 6.2.2.6

- The software criticality classification must be **confirmed** at each milestone

# Software dependability and safety (6/7)

## 6.2.2.4

- **The supplier must apply engineering measures to**
  - reduce the number of critical software components
  - mitigate the risks associated with the critical software (⇒ subclause 6.2.3)



## 6.2.2.7

- **Results of software level analyses are fed back to be integrated into system level analyses**
  - additional software failure modes identified
  - recommendations for the system
  - e.g. introduction of hardware inhibits, modification of system architecture

# Software dependability and safety (7/7)

6.2.8

## ▪ Contribution of software to **Hardware-Software Integration Analysis**

- Identify, for each hardware failure included in the HSIA, the requirements that specify the software behaviour in the event of that hardware failure

6.2.9

## ▪ Verification of **Hardware-Software Integration Analysis** requirements

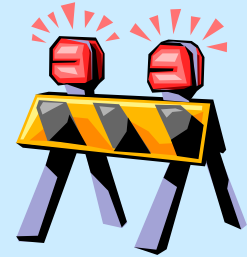
- The software must react correctly to hardware failures
- No undesired software behaviour, that may lead to system failures



# Handling of critical software (1/2)

## 6.2.3.1

- **Propagation** of failures between SW components of different criticality must be prevented
  - If impossible, all interested components classified to the same (**highest**) **criticality**



## 6.2.3.2

- The supplier must define, justify and apply **measures** to assure the dependability and safety of critical software
  - a **set of measures** is proposed



## 6.2.3.3

- The correct implementation of the chosen measures must be **verified** and **reported on**

# Handling of critical software (2/2)

## Specific requirements for critical software

6.2.3.4

- Mandatory **regression testing** in case of change of hardware or development tools

6.2.3.5

- Potential need for **additional verification and validation** to be analyzed in case of change of hardware and environment

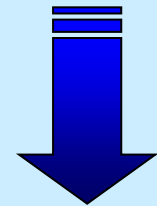
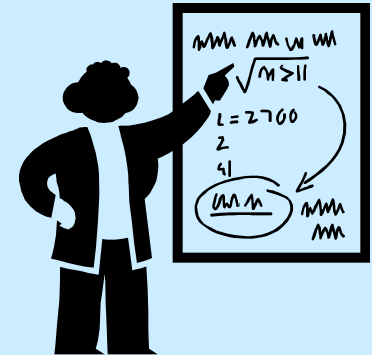
6.2.3.6

- Remove **unreachable code**

6.2.3.7

- Testing to be (re-)executed on **non-instrumented code**

6.2.3.8



# Software configuration management (1/2)

## 6.2.4.1

- **ECSS-M-ST-40** applies to software configuration management
- **Additional requirements in ECSS-Q-ST-80 for assurance aspects**

## 6.2.4.2

- It must be possible to regenerate any software reference version from back-ups

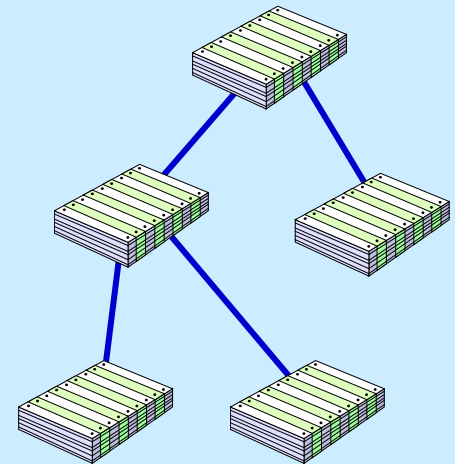
## 6.2.4.3

- Requirements for the release of software configuration file and software release document

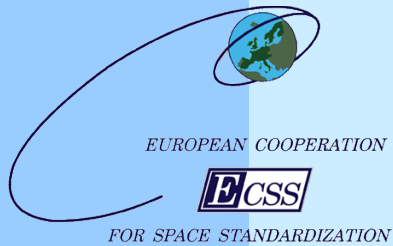
## 6.2.4.4

- Verify that authorized changes are implemented as specified in the CM plan (PA task)

## 6.2.4.6







# Software configuration management (2/2)

6.2.4.7

- **Configuration control** must be applied to
  - Customizable code generation tools (e.g. compilers)
  - Planning documents (e.g. development plan)
  - Development documents (e.g. test specifications)

6.2.4.5

- The **change control procedures** must address the customization of code generation tools

6.2.4.8

- Methods and tools must be identified and applied to **prevent supplied software corruption**

6.2.4.9

- **Checksum** on operational software

6.2.4.11

- **Marking of delivered media**

# Process metrics

6.2.5.1

6.2.5.2

- The supplier must collect, store and analyze **process metrics**, to assess the quality of the development processes
  - Process metrics are based on **quality models** (5.2.7)
- Mandatory **basic metrics** to be
  - used **internally** (duration and effort)
  - reported **to customer** (problems during V&V)
- The supplier must include **process metrics reports** in the software product assurance reports

6.2.5.3

6.2.5.4

6.2.5.5



# Verification (1/5)

## 6.2.6.1

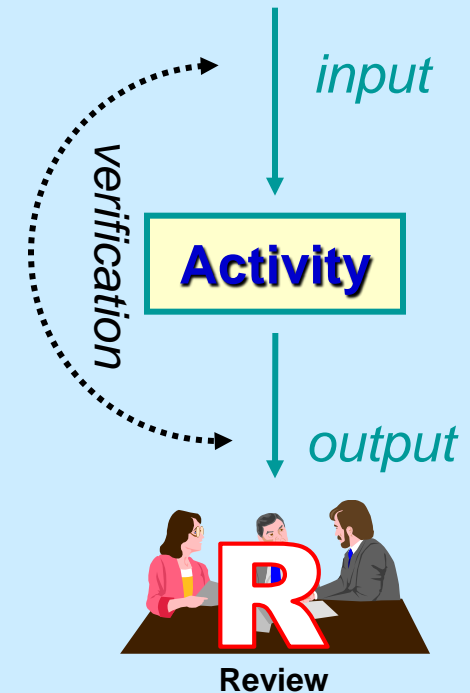
- **Verification of quality requirements** must be planned for

- **Verification includes various techniques**
  - Basically paperwork, not testing

- **The outputs of each activity must be verified against pre-defined criteria**

## 6.2.6.2

- **Only outputs successfully verified can be used in subsequent activities**



# Verification (2/5)

6.2.6.7

- The supplier must **ensure** that
  - adequate verification activities are planned for
  - verification is carried out in accordance with the planning

6.2.6.4

- The **completion of actions** linked to software problem reports generated during verification must be checked



6.2.6.3

- **Assurance activities** on verification and relevant findings must be reported on

# Verification (3/5)

## ▪ Specific verification required for

- deactivated code  $\Rightarrow$   
no accidental or harmful activation

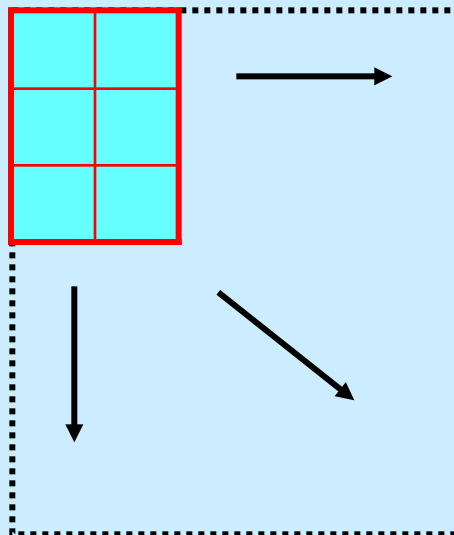
```

...
#ifdef BUS_1553
    /* deactivated */
#endif
...

```

6.2.6.5

6.2.6.6



- configurable code  $\Rightarrow$   
no unintended configuration included in executable or activated at run-time

# Verification (4/5)

## 6.2.6.8



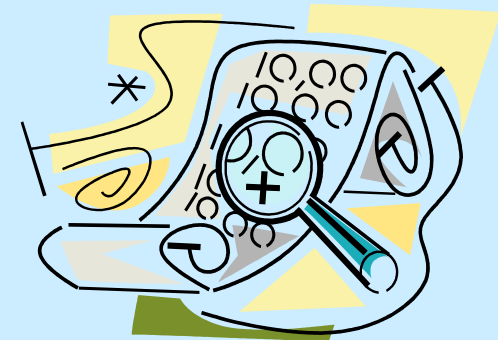
- **Reviews and inspections must be**
  - carried out according to defined **criteria**
  - performed by suitably **independent** personnel
  - based on **written procedures**
  - **reported** on



## 6.2.6.11

## 6.2.6.12

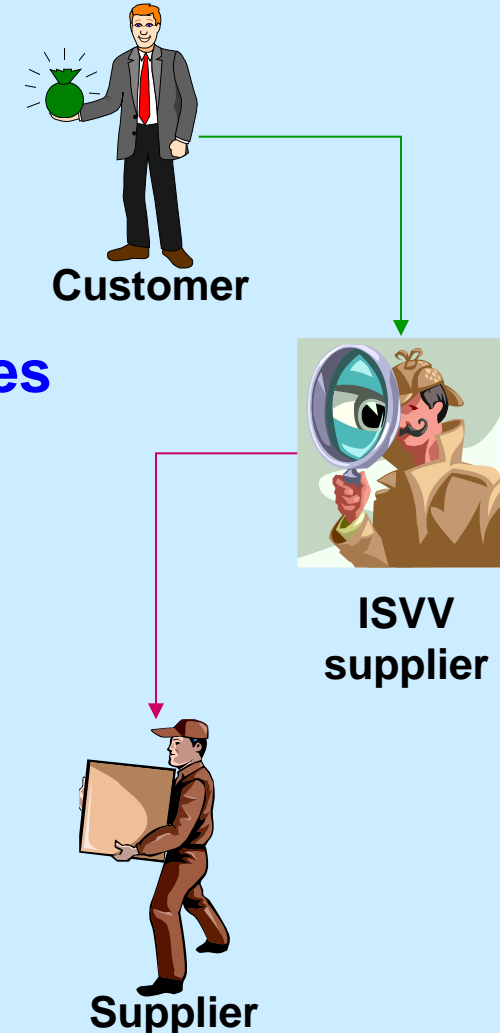
- **Verification of **traceability matrices** must be performed at each milestone review**



**6.2.6.13**

- **Independent software verification** must be performed by a **third party**
- **ISVV** includes several **techniques**
  - review
  - inspection
  - analysis
  - ...
- **Applicability** and **level of independence** to be considered based on **project risks...**

# Verification (5/5)



# Reuse of existing software (1/2)



6.2.7.2

- **Choice: reuse or develop from scratch?**
- **Analysis based on:**
  - assessment of existing software w.r.t. **applicable requirements**
  - evaluation of **quality status** of the existing software, including detailed information about the *documentation status, test coverage, nonconformances, performance, etc.*
  - **other aspects**, such as *warranty conditions, support documentation, conditions of installation and use, intellectual property rights, licencing, etc.*
- **The supplier must document the reuse analysis results in a **software reuse file****

6.2.7.6



## Reuse of existing software (2/2)

6.2.7.5

- The software reuse file must include an estimation of the **level of reuse**

6.2.7.7

- In case the software proposed for reuse does not meet the project requirements, the software reuse file must document the identified **corrective actions**, which can include

- reverse engineering
- delta verification and validation
- documentation of **product service history**



6.2.7.9

- The software reuse file must be submitted to the **customer for approval** and updated at milestones to reflect **corrective actions implementation**

# Requirements applicable to individual SW engineering processes/activities

## Software product assurance

- 5.1 Organization and responsibilities
- 5.2 Software product assurance programme management
- 5.3 Risk management and configuration item control

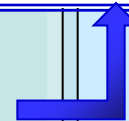
## Software process assurance

- 6.1 Software development life cycle
- 6.2 Requirements applicable to all software engineering processes
- 6.3 Requirements applicable to individual software engineering processes or activities

## Software product quality assurance

- 7.1 Product quality objectives and metrication
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

- 6.3.1 Software related system requirements process
- 6.3.2 Software requirements analysis
- 6.3.3 Software architectural design and design of software items
- 6.3.4 Coding
- 6.3.5 Testing and validation
- 6.3.6 Software delivery and acceptance
- 6.3.7 Operations
- 6.3.8 Maintenance



# Software related system requirements process

6.3.1.1

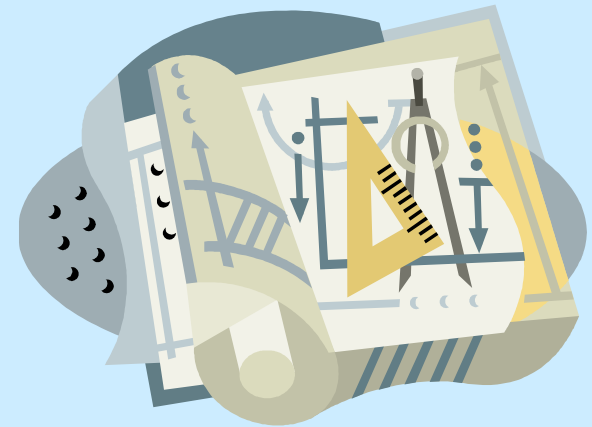
- The requirements for this process are defined in **ECSS-E-ST-40C**, subclause 5.2

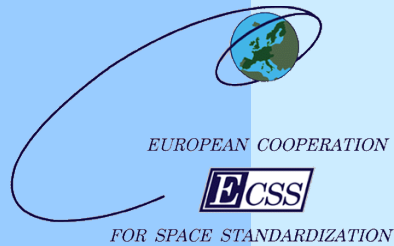
6.3.1.2

- The requirement baseline must be kept under **configuration control**

6.3.1.3

- For the definition of the system requirements applicable to software, the **results of the system level safety and dependability analysis** must be used





# Software requirements analysis

6.3.2.1

- The software requirements must be **fully and unambiguously** defined in the TS

- derived from **requirement baseline**
- using results of **RAMS analyses**
- kept under **configuration control**

6.3.2.2

6.3.2.3

6.3.2.4

- **Non-functional requirements** must be specified
- **Customer and supplier must agree upon**

6.3.2.5

- Responsibility for the TS (on both sides)
- Methods for agreeing on requirements and changes
- Effort to prevent misunderstandings (e.g. dictionary)

# Software architectural design and design of software items

6.3.3.2

- The supplier must define and apply mandatory and advisory **design standards**

6.3.3.3

- including rules for **numerical accuracy**

6.3.3.5

- **Means, criteria and tools** to ensure that the design meets the **quality requirements** must be identified

6.3.3.4

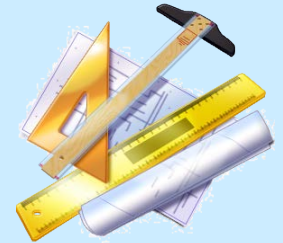
- The supplier must **verify** and **report** on the correct applications of design standards

6.3.3.6

- evaluation to be **fed back** to the design team during the development, for improvement

6.3.3.7

- The design documentation must be suitable for software **maintenance**





# Coding

6.3.4.1

6.3.4.4

6.3.4.3

6.3.4.6

6.3.4.5

6.3.4.8

- **Coding standards must be defined**
  - consistent with the applicable **quality requirements**
  - to be **reviewed** with the customer
  - **measurements, criteria** and **tools** to verify **conformance** of source code with coding standards must be defined in the SPAP
  
- **The supplier must **verify** and **report** on the adherence to coding standards**
  - evaluation to be **fed back** to the programming team during the development, for improvement
  
- **Use of **low-level languages** must be justified**
  
- **Source code under **CM control** after unit tests**

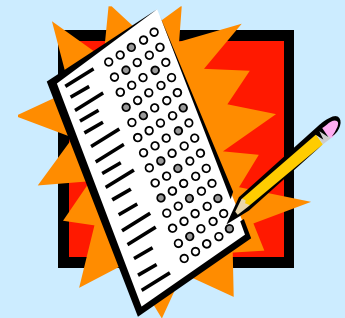
# Testing and validation (1/5)

## 6.3.5.1

- **A testing strategy must be defined and applied**
  - for **each testing level** (*unit, integration, validation against the technical specification, validation against the requirements baseline, acceptance*)
  - **types of tests** (e.g. *functional, boundary, performance, usability*)
  - **product assurance** function involvement

## 6.3.5.2

- **Test coverage goals must be agreed between customer and supplier**
  - based on **criticality** of software
  - at different **testing level** (*unit, integration, etc.*)

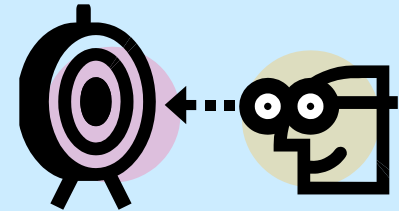


# Testing and validation (2/5)

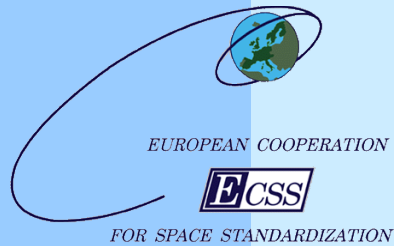
## 6.3.5

### ■ Assurance activities for testing

- verify **suitability, feasibility, traceability, repeatability** of tests
- hold **test readiness reviews**
- check **achievement of test goals**
- allow for **witnessing** of test by PA personnel
- check that the right **software configuration** is tested according to plans and procedure and documented
- **nonconformances and SPRs** must be properly documented
- completion of **actions** deriving from testing nonconformances and SPRs must be verified
- test documentation must be usable for **maintenance**







# Testing and validation (3/5)

6.3.5.15

- **Regression testing** must be performed in case of software modification

6.3.5.16

- documentation must be **updated**

6.3.5.17

- need for regression testing to be evaluated in case of change of **platform hardware** and **code generation tools**

6.3.5.18

6.3.5.19

- **Validation team** must be **different** from development team

6.3.5.26

- The software must be validated as a **whole product**, in an **operationally representative environment**

6.3.5.29

- Test all (or a reasonable number) of the possible **software configurations**

# Testing and validation (4/5)

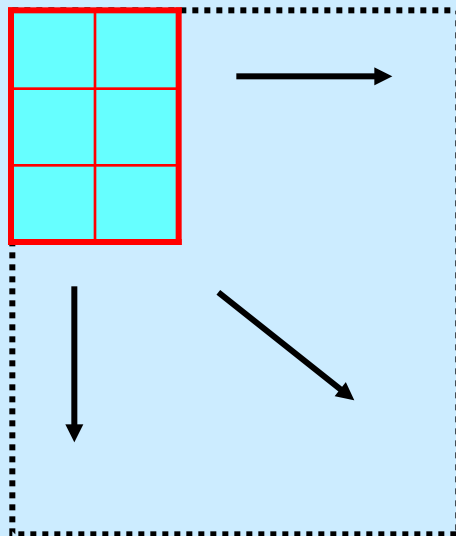
## ■ Specific validation required for

6.3.5.30

- deactivated code  $\Rightarrow$   
no accidental or  
harmful activation

```
...
#ifdef DEBUG
    /* deactivated */
#endif
...
```

6.3.5.31

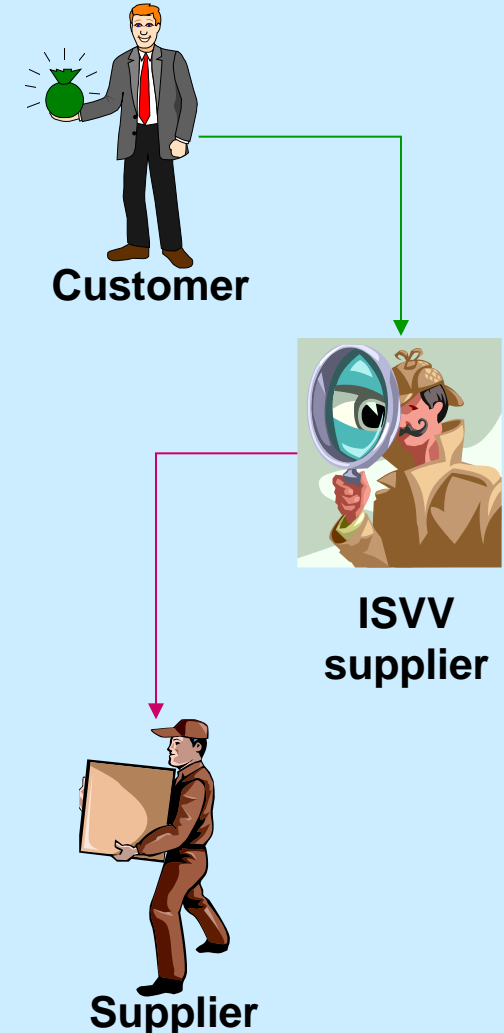


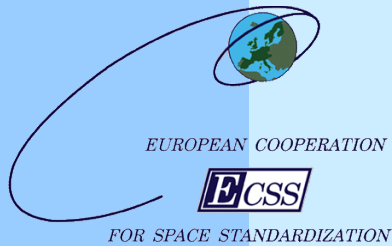
- configurable code  $\Rightarrow$   
no unintended  
configuration included  
in executable or activated  
at run-time

# Testing and validation (5/5)

6.3.5.28

- **Independent software validation** must be performed by a **third party**
- **Applicability** and **level of independence** to be considered based on **project risks...**
- A **less rigorous** level of independence can be considered (e.g. independent team in the same organization)





# Software delivery and acceptance (1/2)

## 6.3.6

- **Installation procedure to be produced**
  - defining **roles and responsibilities** on both sides
- **Acceptance test plan established by the customer**
  - tests from previous phases can be **reused**
- **The supplier must**
  - ensure that the delivered software meets the **contractual requirements**
  - the **source and object code** are the right ones
  - **agreed changes** are implemented
  - **NCRs** are either resolved or declared

# Software delivery and acceptance (2/2)

## 6.3.6

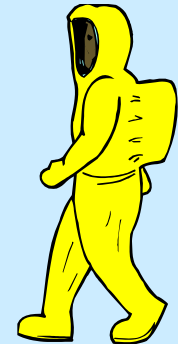
- **Problems during acceptance must be documented in NCRs**
- **The customer must ensure that**
  - the **executable** is generated from controlled code and installed according to installation procedures
  - the tests are executed in accordance with the **plan**
- **An acceptance report must be produced and signed by both parties**
- **The customer shall state the acceptance tests result (accepted, conditionally accepted, rejected)**



# Operations

## 6.3.7

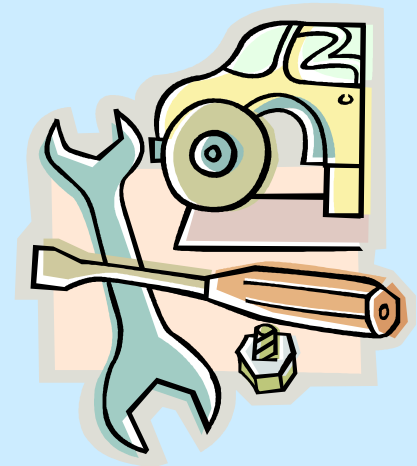
- During operations, the **quality of the mission products** related to software must be agreed upon
- **Validation of the operational requirements**
  - availability and maintainability
  - safety features
  - HMI
  - operational procedures
  - conformance to quality requirements
- **Product assurance plan for operations must consider software operations**

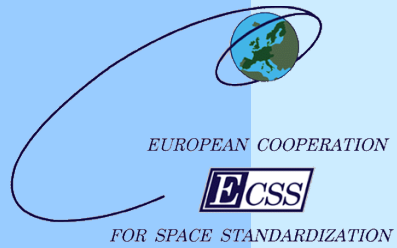


# Maintenance

## 6.3.8

- The **maintenance organization** must be identified early in the life cycle
- A **maintenance plan** must be produced
  - scope of maintenance
  - activities
  - quality measures
  - reporting, ...
- **Records** must be generated for each maintenance activity
  - problems
  - responsibilities
  - solutions, ...



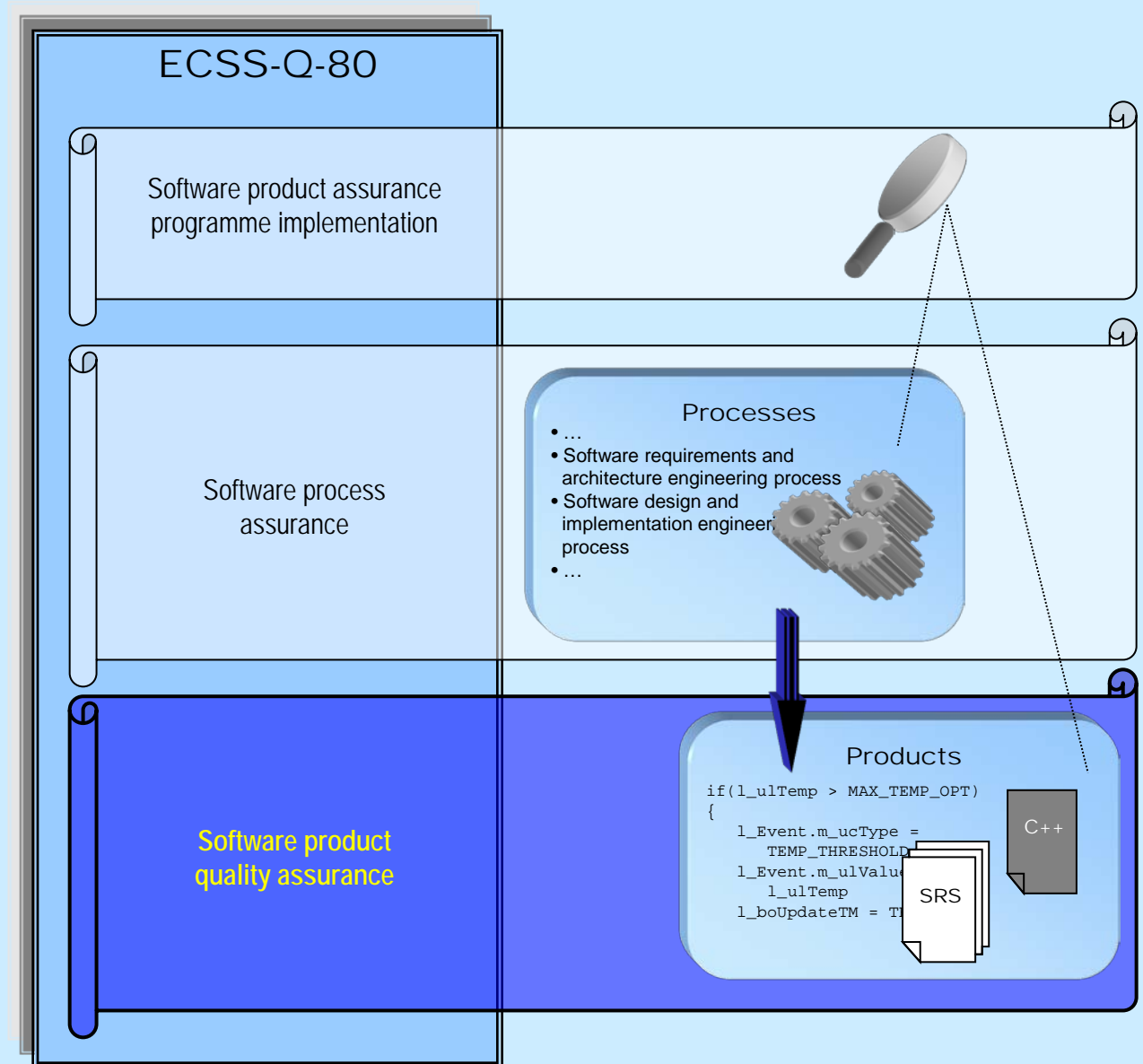


# Software Product Quality Assurance



# Ensuring the quality of the product

**Clause 7**



# Product quality objectives and metrication

## Software product assurance programme implementation

- |                                     |                                    |
|-------------------------------------|------------------------------------|
| 5.1 Organization and responsibility | 5.4 Supplier selection and control |
| 5.2 Software product assurance      | 5.5 Procurement                    |
| 5.3 Risk management programme       |                                    |
| 5.3 Risk management item control    |                                    |

## Software product quality assurance

- 6.1 Software product quality assurance  
6.2 Requirements management  
6.3 Requirements management or activities

- 7.1.1 Deriving of requirements
- 7.1.2 Quantitative definition of quality requirements
- 7.1.3 Assurance activities for product quality requirements
- 7.1.4 Product metrics
- 7.1.5 Basic metrics
- 7.1.6 Reporting of metrics
- 7.1.7 Numerical accuracy
- 7.1.8 Analysis of software maturity

## Software product quality assurance

- 7.1 Product quality objectives and metrication**
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

# Product quality requirements

7.1.1

- **Software quality requirements** (including safety) derive from system requirements

7.1.2

- Quality requirements must be expressed in **quantitative** terms
  - As far as possible, at least...

7.1.3

- The software product assurance function must ensure that quality requirements are documented in the **technical specification**
  - Not only the SPAP



# Product metrics

## 7.1.4

- The supplier must define a **metrication programme** to verify the implementation of quality requirements
  - metrics to be collected
  - target values
  - analyses to be performed
  - usage of metrics
  - schedule of collection



## 7.1.5

- Mandatory **basic metrics** must be used
  - Size, complexity, failures, test coverage

## 7.1.6

- Metrics must be **reported** regularly

# Product quality requirements

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

## Software process assurance

- 6.1 Software development life cycle
- 6.2 Requirements
- 6.3 Requirements or activities

- 7.2.1 Requirements baseline and technical specification
- 7.2.2 Design and related documentation
- 7.2.3 Test and validation documentation

## Software product quality assurance

- 7.1 Product quality objectives and metrics
- 7.2 Product quality requirements**
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

# Requirements baseline and technical specification

## 7.2.1.1

- **Software quality requirements** must be documented in RB and TS

## 7.2.1.2

- **Software requirements must be**
  - correct
  - unambiguous
  - complete
  - consistent
  - verifiable
  - traceable



## 7.2.1.3

- **For each requirement the method for verification and validation shall be specified.**

# Design and related documentation

## 7.2.2

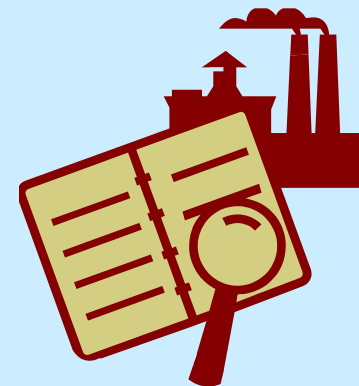
- The software design must meet **non-functional requirements**
- The software design must **facilitate testing**
- Software with a **long lifetime** must be designed so to be independent from operating system and hardware
  - Potential **obsolescence** and non-availability of operational environment



# Test and validation documentation

## 7.2.3

- **The test documentation must**
  - cover test environment, tools, personnel, training needs
  - criteria for test completion and contingency steps
  - test procedures, data and expected results
  - hardware and software configuration
- **For any requirements not covered by testing, a report must be produced to document the verification activities performed**





# Software intended for reuse

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

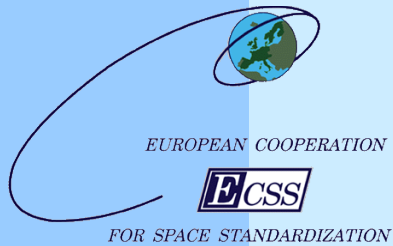
## Software product requirements

- 6.1 Software requirements
- 6.2 Requirements for reuse
- 6.3 Requirements for reuse or activation

- 7.3.1 Customer requirements
- 7.3.2 Separate documentation
- 7.3.3 Self-contained information
- 7.3.4 Requirements for intended reuse
- 7.3.5 Configuration management for intended reuse
- 7.3.6 Testing on different platforms
- 7.3.7 Certificate of conformance

## Software product quality

- 7.1 Product quality objectives and metrics
- 7.2 Product quality requirements
- 7.3 Software intended for reuse**
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware



# Software intended for reuse

## 7.3

- The **documentation** of software intended for reuse in the technical specification design justification file, design definition file and product assurance file must be
  - **separated** from the other
  - **self-contained**
- Requirements for **maintainability, portability and verification** in TS
- Specific **configuration management**
- Testing on all **target platforms**

# Standard ground hardware and services for operational system

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

## Software process assurance

- 6.1 Software development life cycle
- 6.2 Requirements
- 6.3 Requirements or activities

- 7.4.1 Hardware procurement
- 7.4.2 Service procurement
- 7.4.3 Constraints
- 7.4.4 Selection
- 7.4.5 Maintenance

## Software product

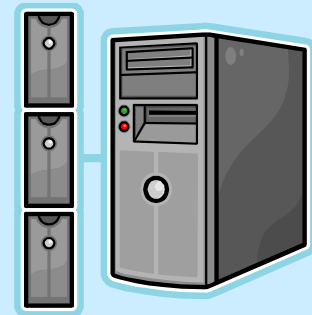
- 7.1 Product quality requirements
- 7.2 Product quality requirements
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

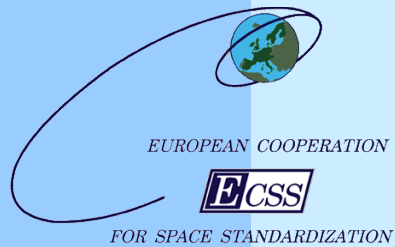


# Standard ground hardware and services for operational system

## 7.4

- Procurement of **operational hardware** to be made in accordance with ECSS-Q-ST-20C, subclause 5.4 (typo)
- Justification of procurement of **operational services** must be provided
- Selection of hardware and services based on specified **criteria** and taking into account development and operational **constraints**
- **Maintenance** of the operation hardware and services must be ensured throughout the entire software operational life





# Standard ground hardware and services for operational system

## Software product assurance programme implementation

- |   |  |
|---|--|
| 5.1 Organization and responsibility                 | 5.4 Supplier selection and control     |
| 5.2 Software product assurance programme management | 5.5 Procurement                        |
| 5.3 Risk management and critical item control       | 5.6 Tools and supporting environment   |
|   | 5.7 Assessment and improvement process |

## Software process assurance

- 6.1 Software development life cycle
- 6.2 Requirements applicable to all software engineering processes
- 6.3 Requirements applicable to individual software engineering processes or activities

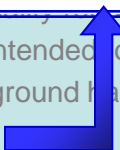
## Software

- 7.1 P
- 7.2 P
- 7.3 Software intended for reuse
- 7.4 Standard ground hardware and services for operational system
- 7.5 Firmware

7.4.1 Device programming

7.4.2 Marking

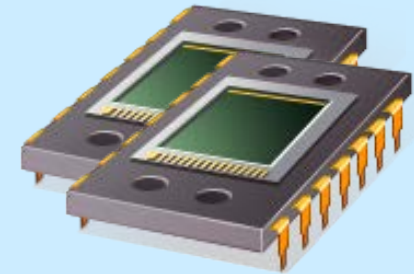
7.4.3 Calibration

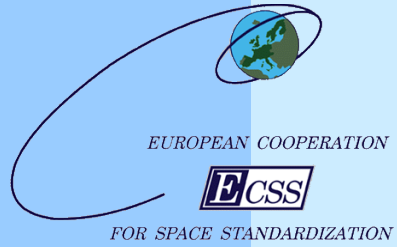


# Firmware

## 7.5

- The supplier must establish procedures for **firmware device programming** and **duplication** of firmware devices.
- Firmware devices must be **marked** for the identification of the hardware and software
- Firmware programming equipment must be **calibrated**

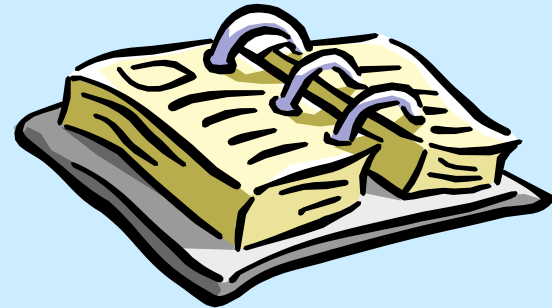




# Handbooks

# Handbooks (1/3)

- Four **ECSS-Q-HB-80-XX** handbooks exist
- Initially drafted in 2006, put on hold and then **updated** to reflect the changes of ECSS-Q-ST-80 from version B to version C
- Handbooks are not Standards, they provide **guidelines** on the application of specific ECSS-Q-ST-80 requirements





# Level-3 Handbooks (2/3)

## ■ **ECSS-Q-HB-80-01 Software Reuse**

- Guidance on how to apply ECSS-Q-ST-80 requirements on reuse of existing software (e.g. subclause 6.2.7)



## ■ **ECSS-Q-HB-80-02 Software Process Assessment and Improvement**

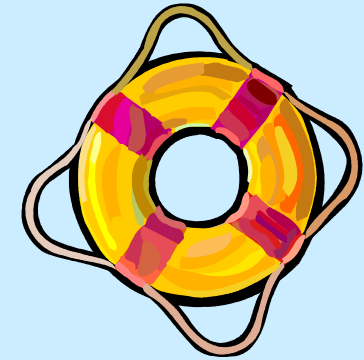
- Guidance on how to apply ECSS-Q-ST-80 requirements on software process assessment and improvement (e.g. subclause 5.7)



# Level-3 Handbooks (3/3)

- **ECSS-Q-HB-80-03 Software Dependability and Safety**

- Guidance on how to apply ECSS-Q-ST-80 requirements on SW RAMS (e.g. subclause 6.2.2)



- **ECSS-Q-HB-80-04 Software Metrication Programme**

- Guidance on how to apply ECSS-Q-ST-80 requirements on software quality models and metrics (e.g. subclauses 5.2.7, 6.2.5, 7.1)

