Block #3 – PA in the Software Life Cycle

ESA UNCLASSIFIED - For ESA Official Use Only

PA in the Software Life Cycle
Agenda

- **SW Engineering Foundation**
  - Plans & Procedures
  - Tools and Development Environment
- **SW Engineering related processes**
  - Walk through V-cycle
  - Involvement of PA
  - Beyond V-cycle
- **SW Reuse**
  - SW life cycles processes
- **SW PA challenges**
  - Automatic code generation
  - AGILE practices
  - Points of Attention

Plans and Procedures

- Software development and operations processes need to be documented in plans and procedures
  - To be controlled and repeatable
- Plans and procedures shall be finalized before the relevant activities start

- Plans shall be reviewed and updated as necessary at each milestone
- Procedure and standards shall be reviewed by all project actors, for suitability and feasibility, and against relevant plans and contractual requirements
  - E.g. Change Control procedure, Coding Standards

Tools and Development Environment (I)

- Methods and tools for:
  - Requirements analysis
  - Software specification
  - Modelling
  - Design
  - Coding
  - Testing
  - Validation
  - Configuration management
  - Verification
  - Product assurance

- … shall be identified and agreed with the customer
- It shall be demonstrated that:
  - The team has sufficient experience to use them
  - Are appropriate for the functional and operational characteristics of the product
  - Are available (in an appropriate hardware environment) throughout the development and maintenance lifetime

ECSS-Q-HB-80-01 Section 6

Tools and Development Environment (II)

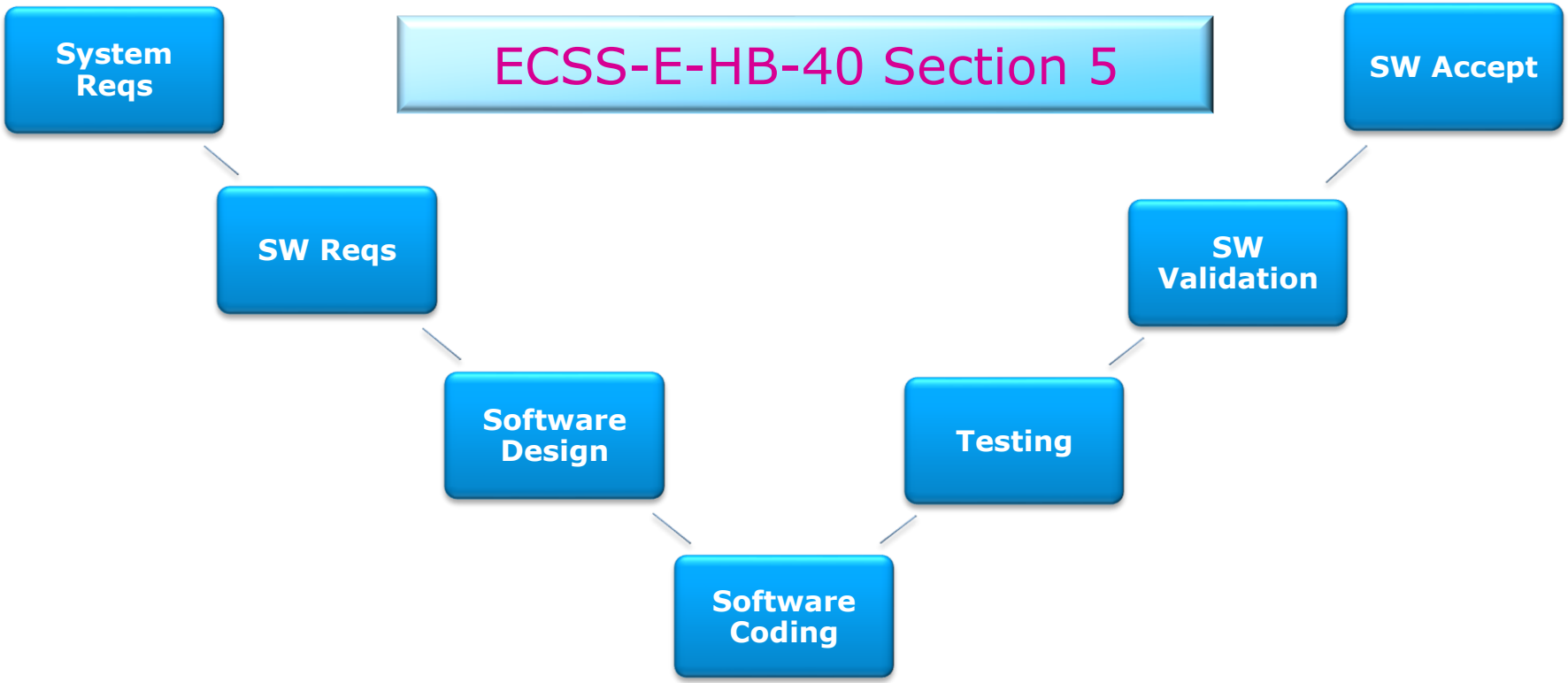
- The development environment shall be selected based on:
  - Availability
  - Compatibility
  - Performance
  - Maintenance
  - Assessment with respect to requirements, including the criticality category
  - *[and many more…]*

- Again, justified and agreed with the customer
- PA responsibility:
  - Ensure availability of tools and environment when needed
  - Report on correct use of methods and tools

Tools and Development Environment (III)

- Capture (in writing!) how you perform
  - tool configuration / setup
  - Usage
  - Data retrieval
  - Data interpretation/analysis
  - Data storage/archiving
  - Reporting
- Why?
  - Think of repeatability
  - Think of handover to a colleague or get support when schedule gets tight

Software Engineering related processes



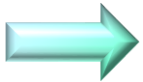
System Requirements for Software

- The Requirement Baseline contains system requirements allocated to software
  - "Derived from an analysis of the specific intended use of the system, and from the results of the safety and dependability analysis" [ECSS-E-ST-40C]
  - Requirements for: functions & performance, V&V, operations, maintenance, in-flight modification, real-time, security, quality, observability, HMI

A thorough and effective requirements elicitation is crucial to the development of software that satisfactorily implements system's and customer's needs

Software Requirements Analysis

- The software requirements need to be fully and unambiguously defined in the Technical Specification
  - Derived from the Requirements Baseline
  - Using results of RAMS analyses
  - Kept under configuration control
- Traceability TS ⇔ RB shall be demonstrated
  - TS requirements not traceable to RB need to be justified
- Non-functional requirements shall be specified
- Test Method shall be defined for each requirement

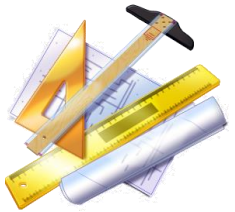


- Performance
- Safety
- Reliability
- Robustness
- Quality
- Maintainability
- Configuration management
- Security
- Confidentiality
- Metrication
- Verification and validation

Software Design

- Mandatory and advisory design standards shall be defined and applied
  - Including rules for numerical accuracy
- Means, criteria and tools to ensure that the design meets the quality requirements shall be identified
- The correct application of design standards shall be verified and reported upon
  - Evaluation to be fed back to the design team during the development, for improvement purposes

The design documentation must be suitable for software maintenance



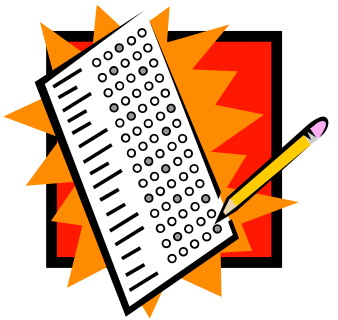
Coding

- Coding standards shall 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 shall be defined
- Use of low-level languages shall be justified
- The source code is to be put under CM control after unit tests
- Adherence to coding standards shall be verified and reported upon

Evaluation to be fed back to the programming team during the development, for improvement purposes

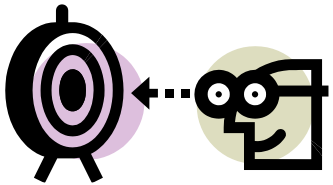
Testing and Validation (I)

- A testing strategy shall 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
- Test coverage goals shall be agreed between customer and supplier
  - Based on criticality of software
  - At different testing level (unit, integration, etc.)
  - *[ECSS-E-ST-40 specifies hard test coverage goals based on criticality]*



Testing and Validation (II)

- 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 shall be properly documented
  - Completion of actions deriving from testing nonconformances and SPRs shall be verified
  - Test documentation shall be usable for maintenance



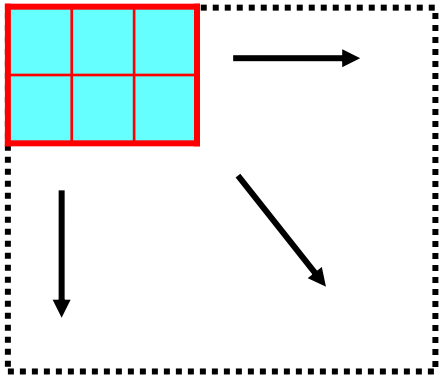
Testing and Validation (III)

- Regression testing shall be performed in case of software modification
  - Documentation shall be updated
  - Need for regression testing to be evaluated in case of change of platform hardware and code generation tools
- Validation team shall be different from development team
- The software shall be validated as a whole product, in an operationally representative environment
- All (or a reasonable number) of the possible software configurations shall be tested

Testing and Validation (IV)

- Specific validation required for:
  - Deactivated code ⇒ avoid accidental or harmful activation

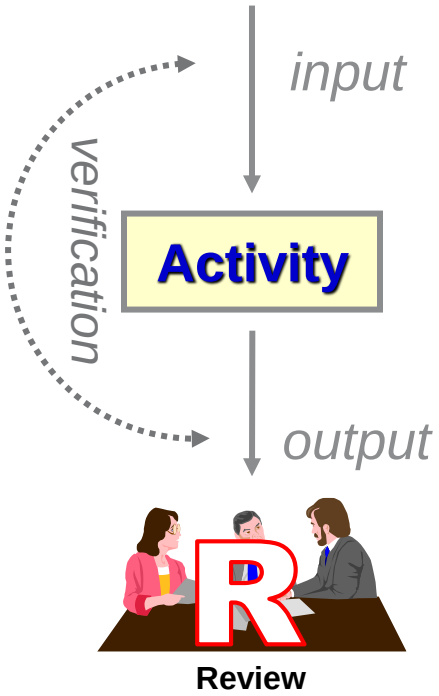
```
...
if(read(PIN_DEBUG) == HIGH)
{
  /* deactivated */
}
...
```



  - Configurable code ⇒ no unintended configuration included in executable or activated at run-time

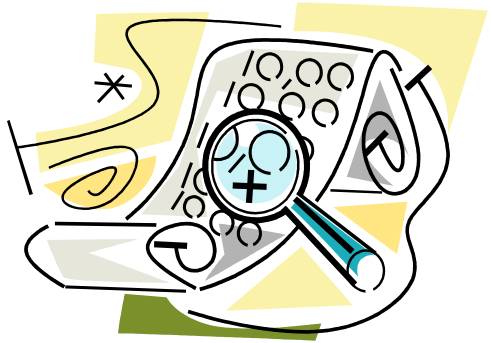
Verification (I)

- Verification includes various techniques
  - Review, inspection, walkthrough, cross-reading, desk-checking, model simulation, …
  - Basically paperwork, not testing
- Verification of quality requirements shall be planned for
- The outputs of each activity shall be verified against pre-defined criteria
- Only outputs successfully verified can be used in subsequent activities



Verification (II)

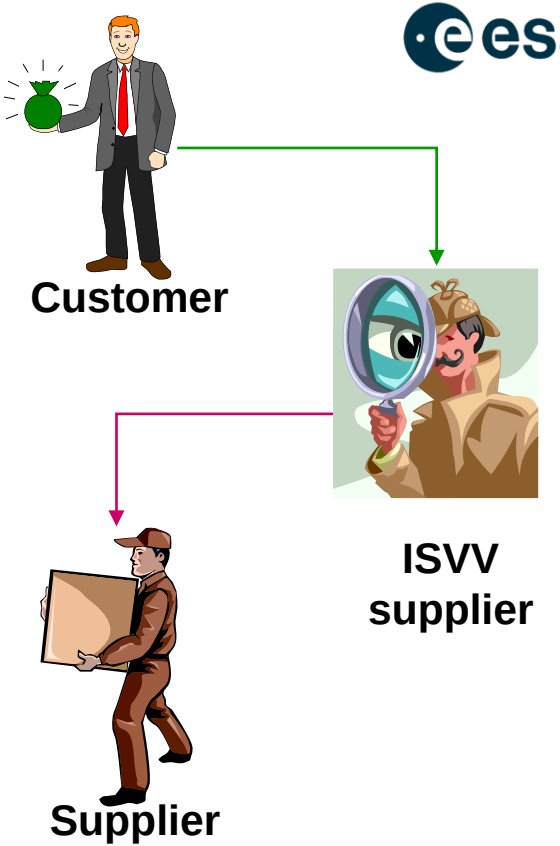
- Reviews and inspections shall be
  - Carried out according to defined criteria
  - Performed by suitably independent personnel
  - Based on written procedures
  - Reported on
- Specific verification shall be carried out for
  - Deactivated code and configurable code
- Verification of traceability matrices shall be performed at each milestone review



ISVV

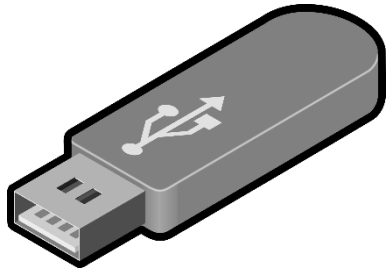
- Independent Software Verification and Validation shall be performed by a third party
  - Combination of reviews, inspections, analyses, simulations, testing and auditing
- *"This requirement is applicable where the risks associated with the project justify the costs involved.*
- *The customer can consider a less rigorous level of independence, e.g. an independent team in the same organization."* [ECSS-Q-ST-80]

(Built-in ECSS-Q-ST-80 tailoring capability)



Software Delivery and Acceptance (I)

- An Installation procedure shall be produced
  - Defining roles and responsibilities on both sides
- An Acceptance test plan shall established by the customer
  - Tests from previous phases can be reused
- The supplier shall:
  - 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

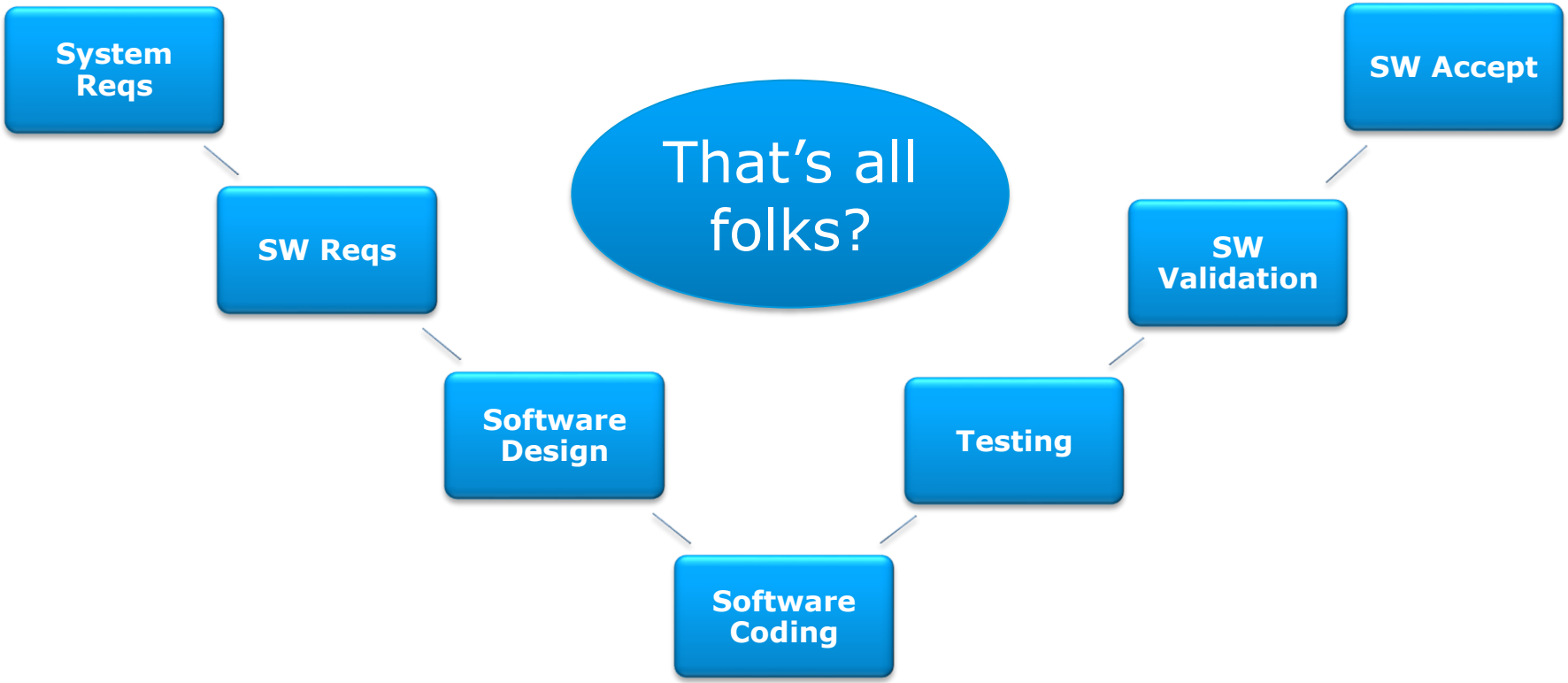


Software Delivery and Acceptance (II)

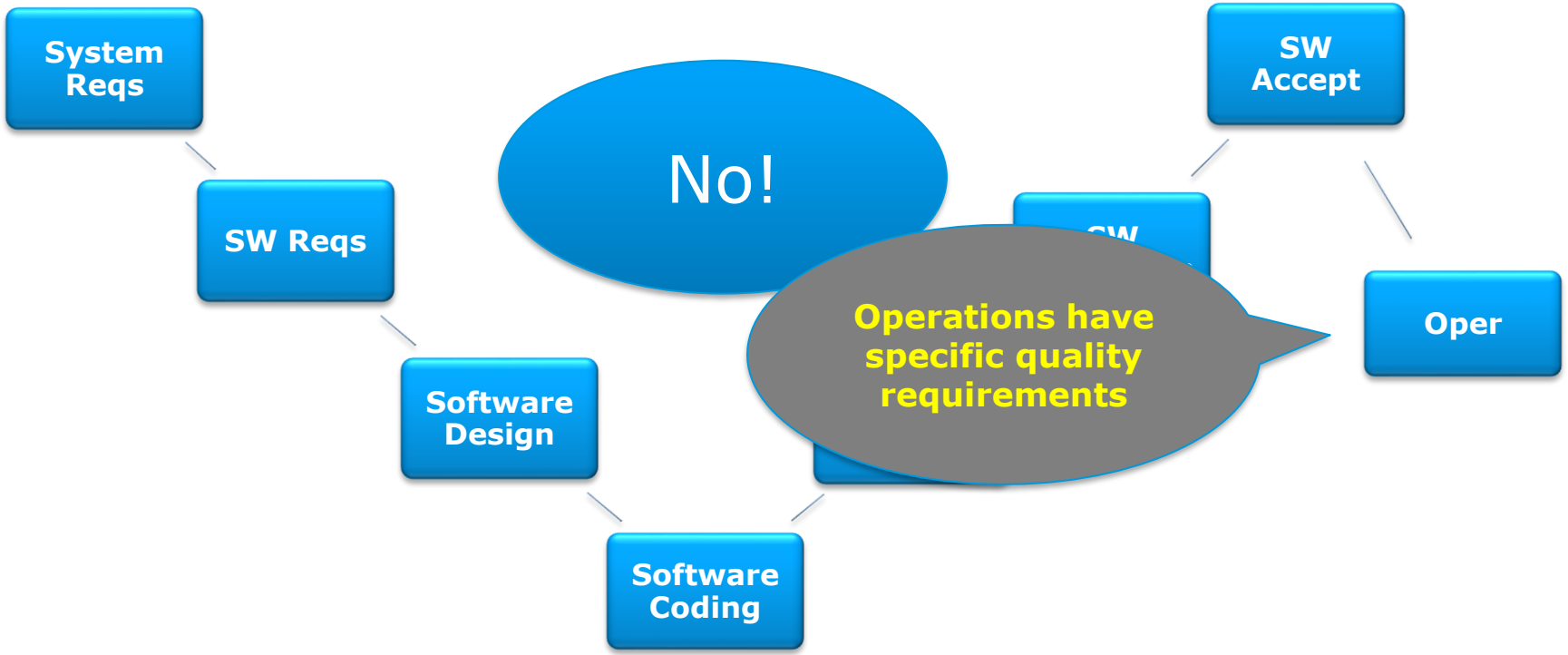
- Problems during acceptance shall be documented in NCRs
- The customer shall ensure that
  - The executable is generated from controlled code and installed according to installation procedures
  - The tests are executed in accordance with the plan
- An acceptance report shall be produced and signed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

SW PA involvement

Software engineering related processes

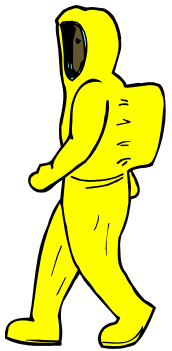


Software engineering related processes

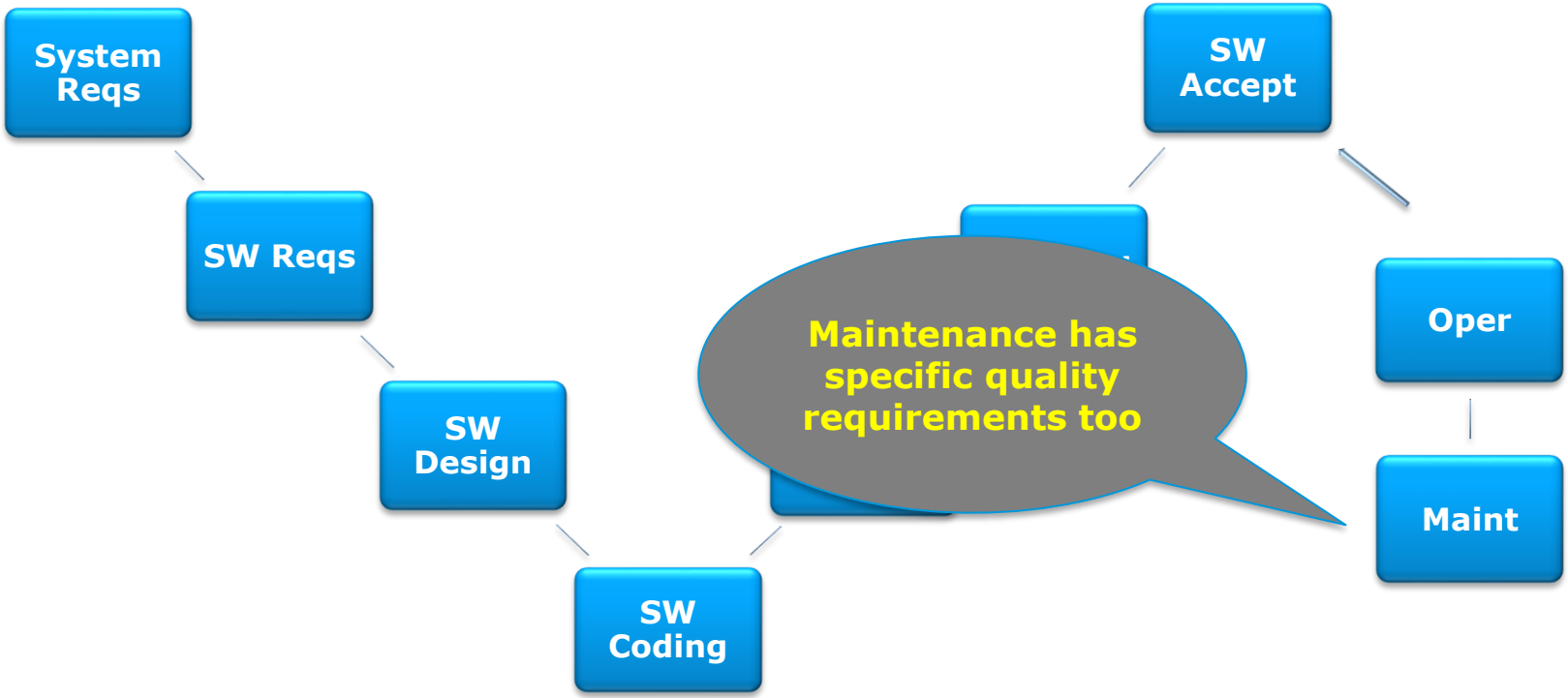


Operations

- Software operations are mostly supported by product assurance activities carried out prior to operations themselves
  - Ensuring software dependability, documentation quality, etc.
- During validation of operations requirements for software, the following shall be addressed, as a minimum
  - Availability and maintainability of the host system
  - Safety features
  - Human-computer interface
  - Operating procedures
  - Ability to meet the mission product quality requirements



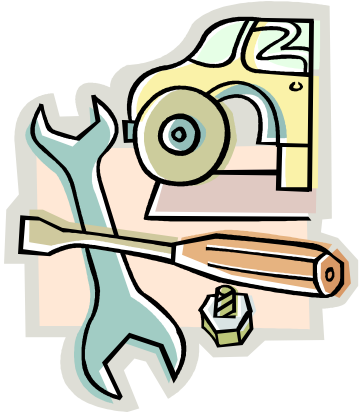
Software engineering related processes



Maintenance (I)

- The maintenance organization shall be identified early in the life cycle
- Maintenance often starts well before operations
  - E.g. for equipment software to be integrated into the system

- A maintenance plan shall be produced
  - Against the specified requirements for maintenance of the software product
  - Addressing assurance, verification and validation activities applicable to maintenance interventions
  - Establishing rules for the submission of maintenance reports



Maintenance (II)

- The maintenance plan shall address
  - Scope of maintenance
  - First version of the software product for which maintenance is to be done
  - Support organization
  - Maintenance life cycle
  - Maintenance activities
  - Quality measures to be applied during the maintenance
  - Maintenance reports
- Detailed maintenance records shall be retained
  - Can be used for enhancement of software products and maintenance system

Reuse of existing software (I)

- Choice: reuse or develop from scratch?
- Analysis of advantages of reuse vs. new development 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, residual nonconformances, performance, code quality, etc.*
  - other aspects, such as warranty conditions, support documentation, conditions of installation and use, intellectual property rights, licencing, etc.

- The supplier shall document the reuse analysis results in a software reuse file

Reuse of existing software (II)

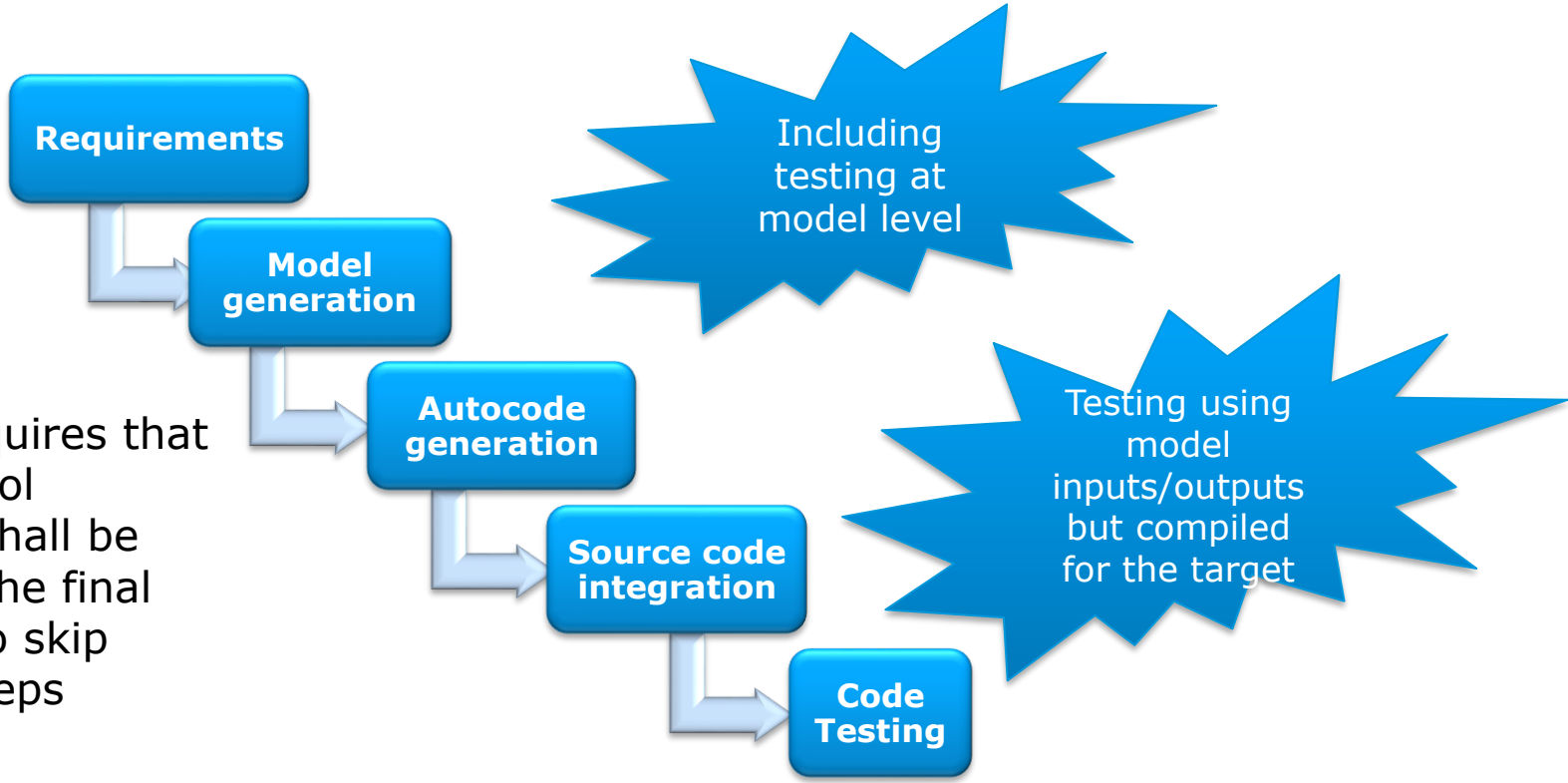
- The software reuse file shall include an estimation of the level of reuse
- In case the software proposed for reuse does not meet the project requirements, the software reuse file must document the identified corrective actions, which can include
  - Reverse engineering
  - Delta verification and validation
  - Documentation of product service history
- The software reuse file must be submitted to the customer for approval and updated at milestones to reflect corrective actions implementation

ECSS-Q-HB-80-01
Reuse of existing software

SW PA challenges

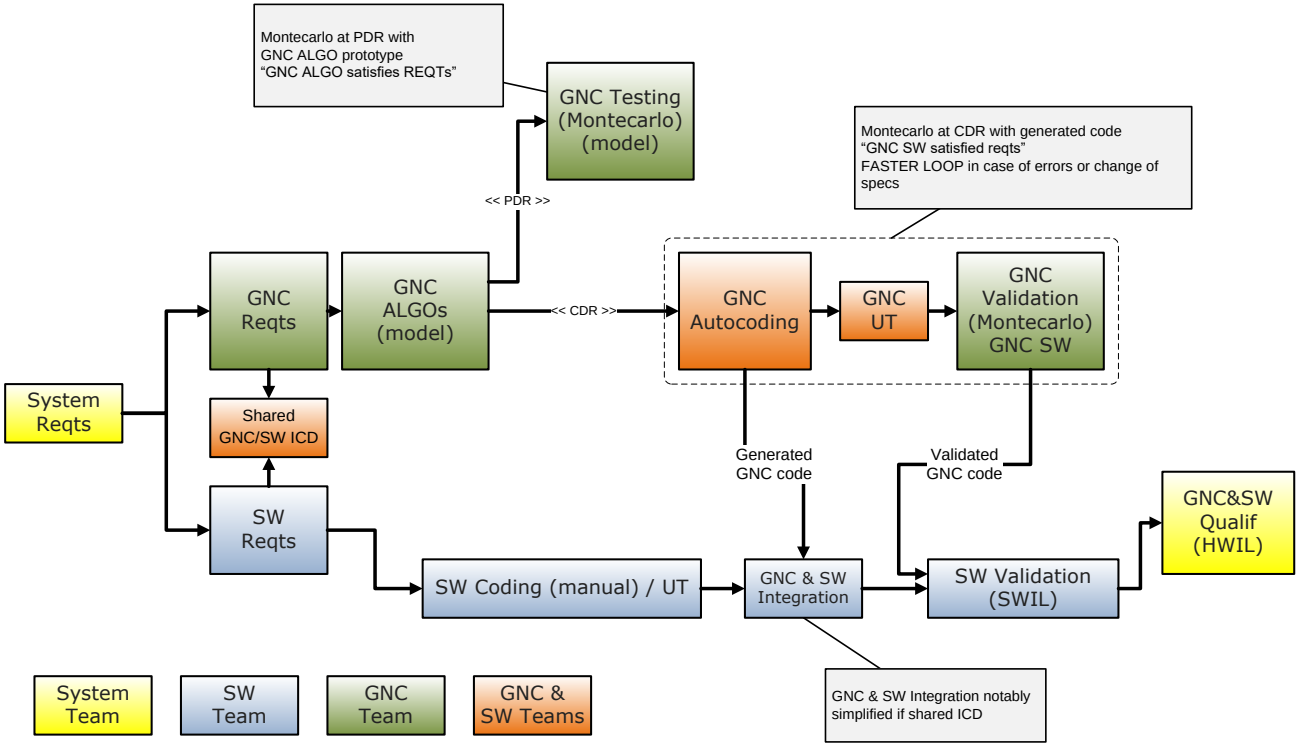
- Automatic code generation
- AGILE practices
- Very Small Entities (VSEs) in space
- FPGA vs SW development
- On-board autonomy

Automatic code generation process



ECSS Q80 requires that the level of tool qualification shall be the same as the final SW product to skip verification steps

Automatic code generation for AOCS



AGILE practices
HB snapshot: Guidelines for Software Engineering Processes

change as a natural part of the life cycle rather than an anomaly

Sprint Demo meetings can be considered as delta CDRs, if the validation of user stories are systematically demonstrated and verified

The product backlog is the repository of the user stories and tasks and can be best mapped to the RB and Software Requirements Specification SRS

ECSS-E-HB-40-01
Agile SW development HB

Agile practices
HB Snapshot: Guidelines for Software Quality Management

Quality Manager as Scrum Master

Critical SW development is supported by the definition of done

providing an independent view at the iteration reviews and retrospectives

Consider using the backlog as the basis for traceability (if feasible) and not introduce yet another document.

The retrospectives are the perfect occasion to assess the process and seek improvement opportunities.

ECSS-E-HB-40-01
Agile SW development HB

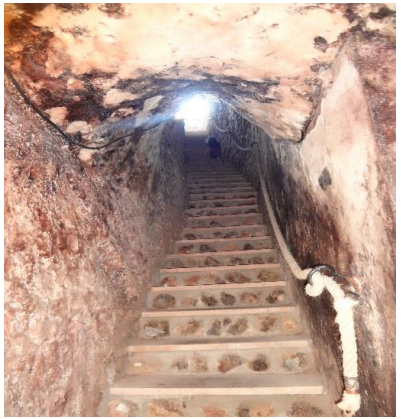
Agile Myths (or misconceptions): debunking

- ✓ Myth 0: You are agile® if you follow all practices if not, you are waterfalle®
- ✓ Myth 1: Agile is better (or worse) than waterfall
- ✓ Myth 2: Agile means no planning
- ✓ Myth 3: Agile means less discipline
- ✓ Myth 4: Agile means no documentation
- ✓ Myth 5: Agile means more flexibility, but less stability

EUROPEAN COOPERATION ECSS FOR SPACE STANDARDIZATION

ECSS Myths (or misconceptions): debunking

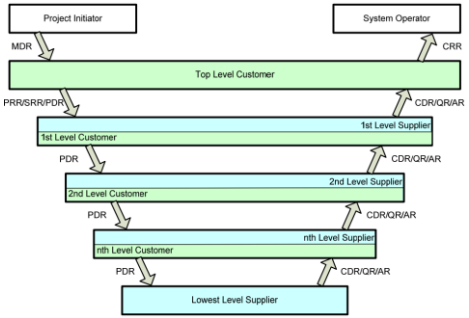
- ✓ Myth 1: E40C/Q80C are pure waterfall.
  - ✓ E40C/Q80C define processes and milestones but other lifecycles are possible.
  - ✓ Iterative Lifecycles using versions are common.
- ✓ Myth 2: E40C/Q80C are very heavy on documentation
  - ✓ E40C/Q80C reqs as many things in ECSS can be tailored (out)
  - ✓ E40C/Q80C DRDs are templates meant to help
- ✓ Myth 3: After SWRR there are no more changes to requirements.
  - ✓ Wrooong!!! Why are we tracking req instability at CDR?
- ✓ Myth 4: E40C/Q80C is only used for flight software
  - ✓ Software Verification Facilities & simulators run on the ground.
- ✓ Myth 5: All flight Software is criticality B at least
  - ✓ Criticality does not come from the environment it runs in.



EUROPEAN COOPERATION FOR SPACE STANDARDIZATION

Points of Attention: Models

- ✓ 5 models are proposed
- ✓ Ranging from:
  - ✓ Model 1:closer to ECSS, but far from Scrum
    - ✓ ECSS-E-ST-40C/ECSS-Q-ST-80C/ECSS-M-ST-10C



ECSS-E-HB-40
6.2.3 and Table 6-1

- ✓ Model 5: closer to Scrum

EUROPEAN COOPERATION
ECSS
FOR SPACE STANDARDIZATION

Points of Attention: Scope & Deadlines

- In a pure agile/Scrum approach the "*requirements are being defined on every sprint*".
- And because the requirements are being defined as they are being developed, "*there is no defined end date*".
- But for some developments at ESA the requirements are known 80% at Kick Off.
- And the whole development & integration of the subsystems/components/elements depend on each other.
  - So **ESA needs DEADLINES that have to be met**.

EUROPEAN COOPERATION
ECSS
FOR SPACE STANDARDIZATION

Points of Attention: Coverage of Flight SW

- A large number of requirements are well-known at Kick-Off.
- Flight Software runs on Flight Hardware that is also being developed.
  - Integration is key
- Flight Software requires a strong degree of formality
  - High criticality
  - Reviews to verify progress so far.

EUROPEAN COOPERATION
ECSS
FOR SPACE STANDARDIZATION

Points of Attention: Visibility

- ✓ Visibility via contractual structure does not reach ESA.
- ✓ Software is developed by a subcontracting company to the PRIME
- ✓ ESA is therefore not the Product Owner
- ✓ But ESA is the final customer so it is a Stakeholder.

- ✓ ESA needs visibility

EUROPEAN COOPERATION ECSS FOR SPACE STANDARDIZATION

Disclaimer

This presentation is a property of the European Space Agency (ESA) or ESA's licensors. No part of this material may be reproduced, displayed, amended, distributed or otherwise used in any form or by any means, without written permission of ESA or ESA's licensors. Any unauthorised activity or use shall be an infringement of ESA's or ESA licensors' intellectual property rights and ESA reserves the right to defend its rights and interests, including to seek for remedies.