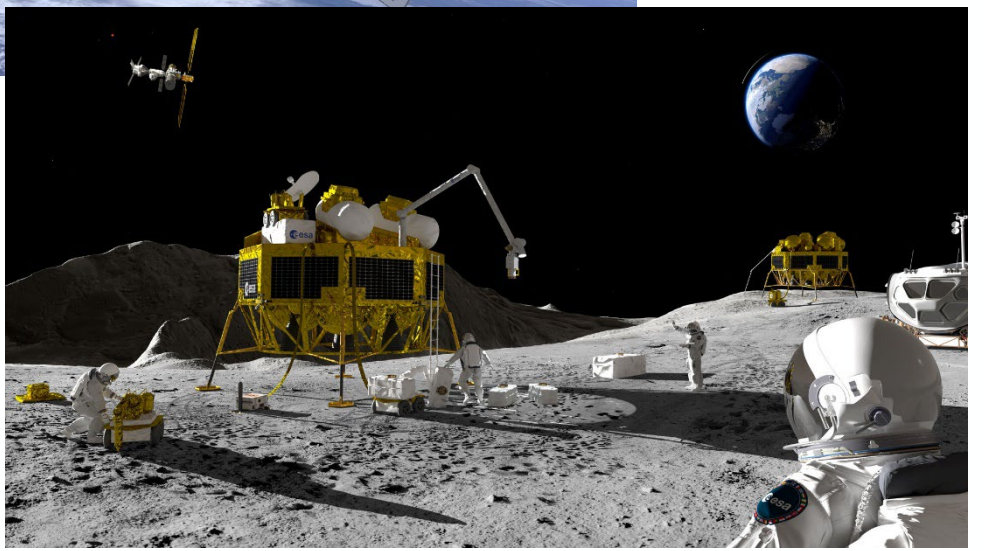
ECSS-Q-ST-80C Rev.1 – Training Course

Block #1 – Software in the Context of Space Projects

30/03/2023

Software in the Context of Space Projects Agenda

- **Introduction**
  - Software in the space system
  - What makes SW for space special
  - Software and Space System Engineering
- **Software in the space project life cycle**
  - Overview from Phase 0 to Phase F
  - SW Reviews versus System Review
- **Space vs ground environment**
  - Space software environments
  - Ground and space segment
- **Software life-cycle & reviews**
  - SW life cycles processes
  - Reviews/DRL/DRDs
  - SW Life cycle models



Software and the space system

- Software implements (more and more of) the **system functionalities/behaviour** (autonomous missions, analogue to digital equipment's)
- System **complexity** increases ➔ software size increases
- **Software schedule** is squeezed within the system schedule
- Software is the last **flexibility** of the system at the end of the life cycle
- Software is **highly flexible** (system level functions to equipment firmware)
- Software touches many parts of the system. It has **interface** everywhere (ground – hardware – avionics – payloads – sensors – actuators – egse – security…)
- Software uses a **lot of data** from various system functional chains (temperature, health status, voltage)
- Software has several **users** (system – testing – operation)
- Software is different from other engineering disciplines (no physical property, no mass nor produces heat..)
- Software engineering is a pure **intellectual activity** **(**principal output is documentation including source code)

What makes SW for space special?

- Embedded software
- Challenging requirements (eg autonomy, time constraints…)
- limited resources (CPU , memory)
- Harsh environment (radiation, thermal, vacuum …)
- Late availability of the final platform
- Difficult access to flight hardware
- Schedule driven by a (rocket) launch date
- Difficult access for SW maintenance (or no access at all)
- SW is mission critical and/or impacting human safety
- Unique development, reuse is difficult
- Long development cycles (5 – 10 years)

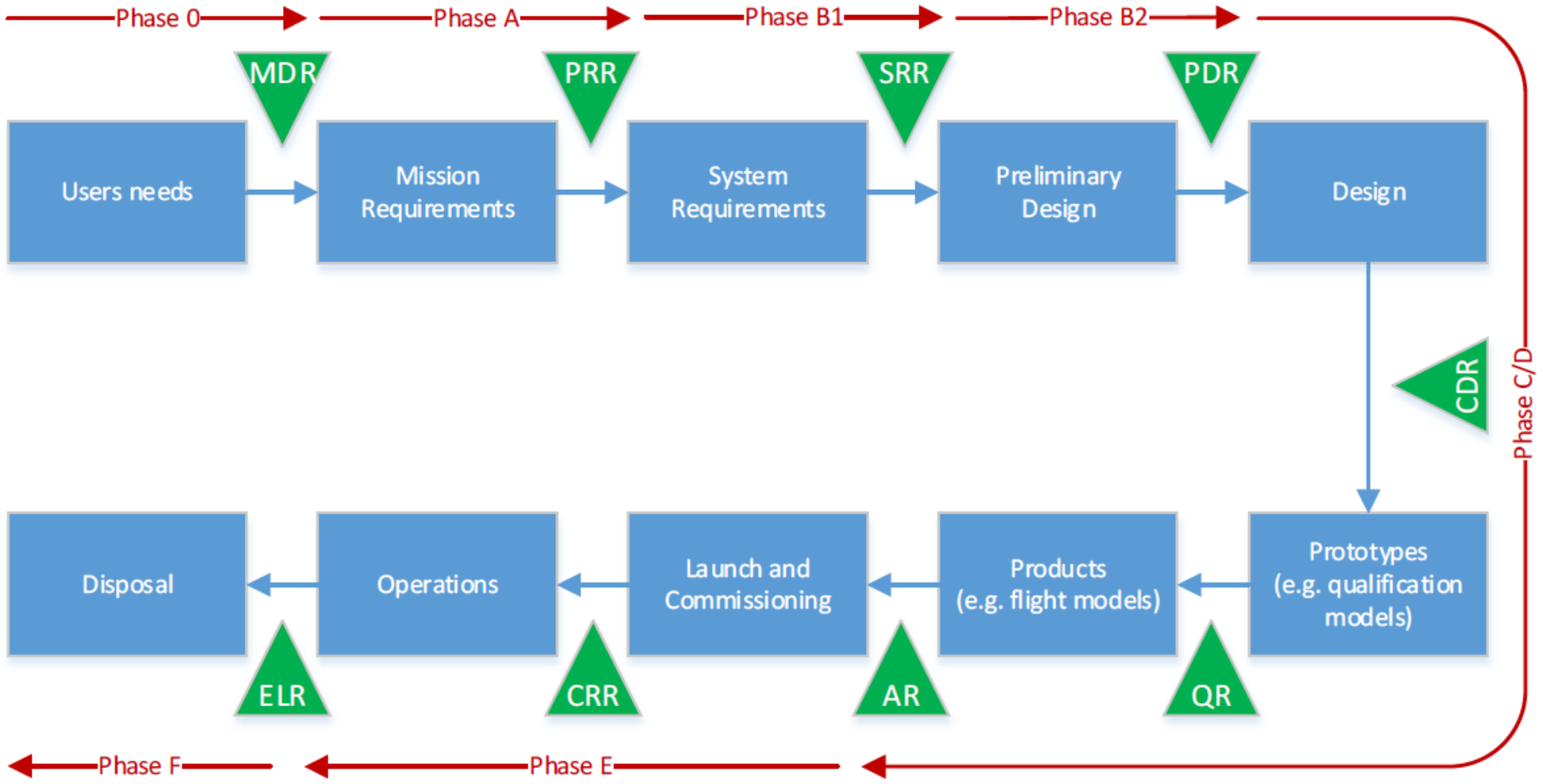
Software and Space System Engineering

The software components of a space system play a role alongside the other engineering components such as mechanical and electrical

All of these various engineering components (including software) are governed by the overall discipline known as **space system engineering**

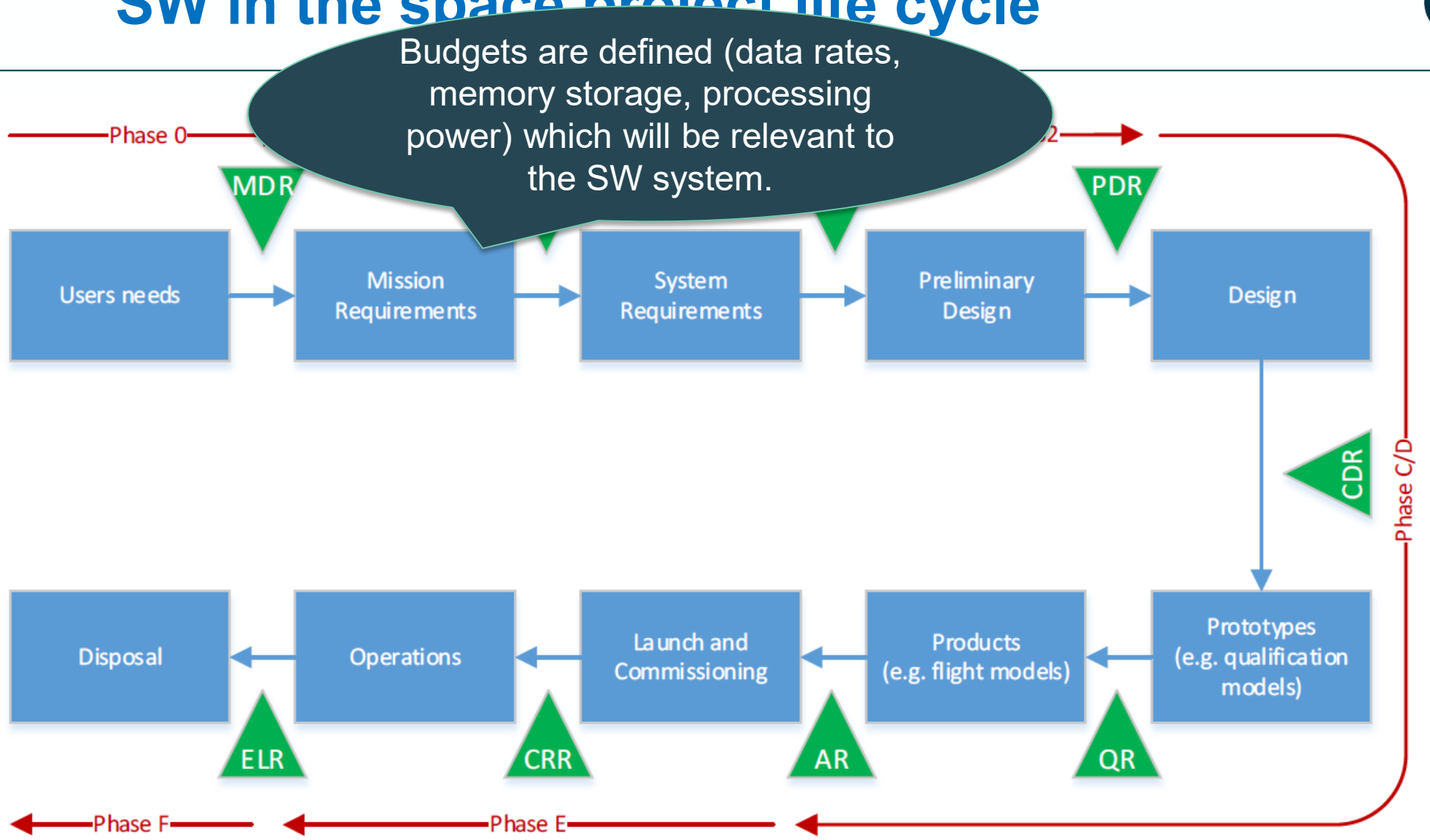
Software components are part of the overall mission system, together with other engineering components

SW in the space project life cycle



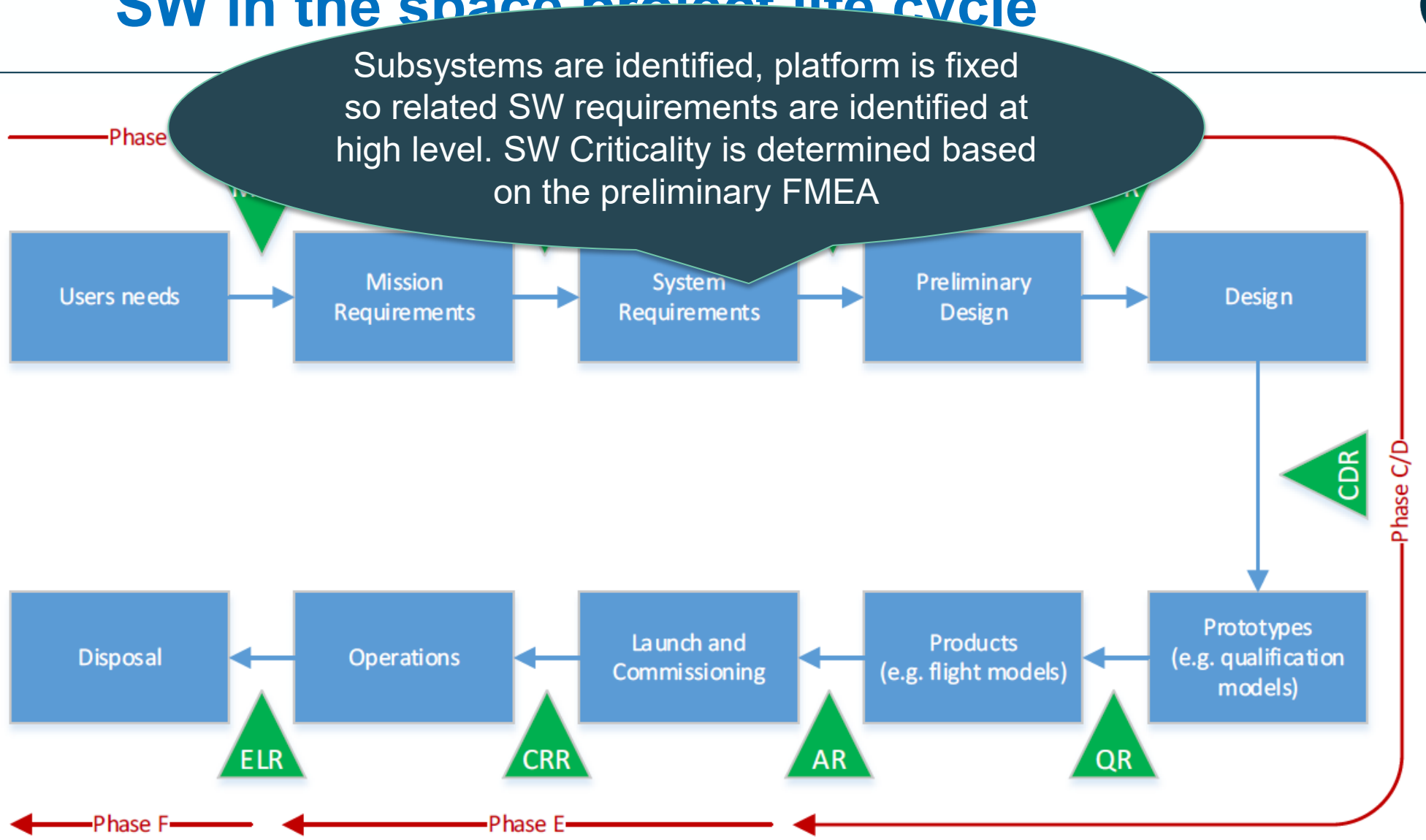
Project Phases

SW in the space project life cycle



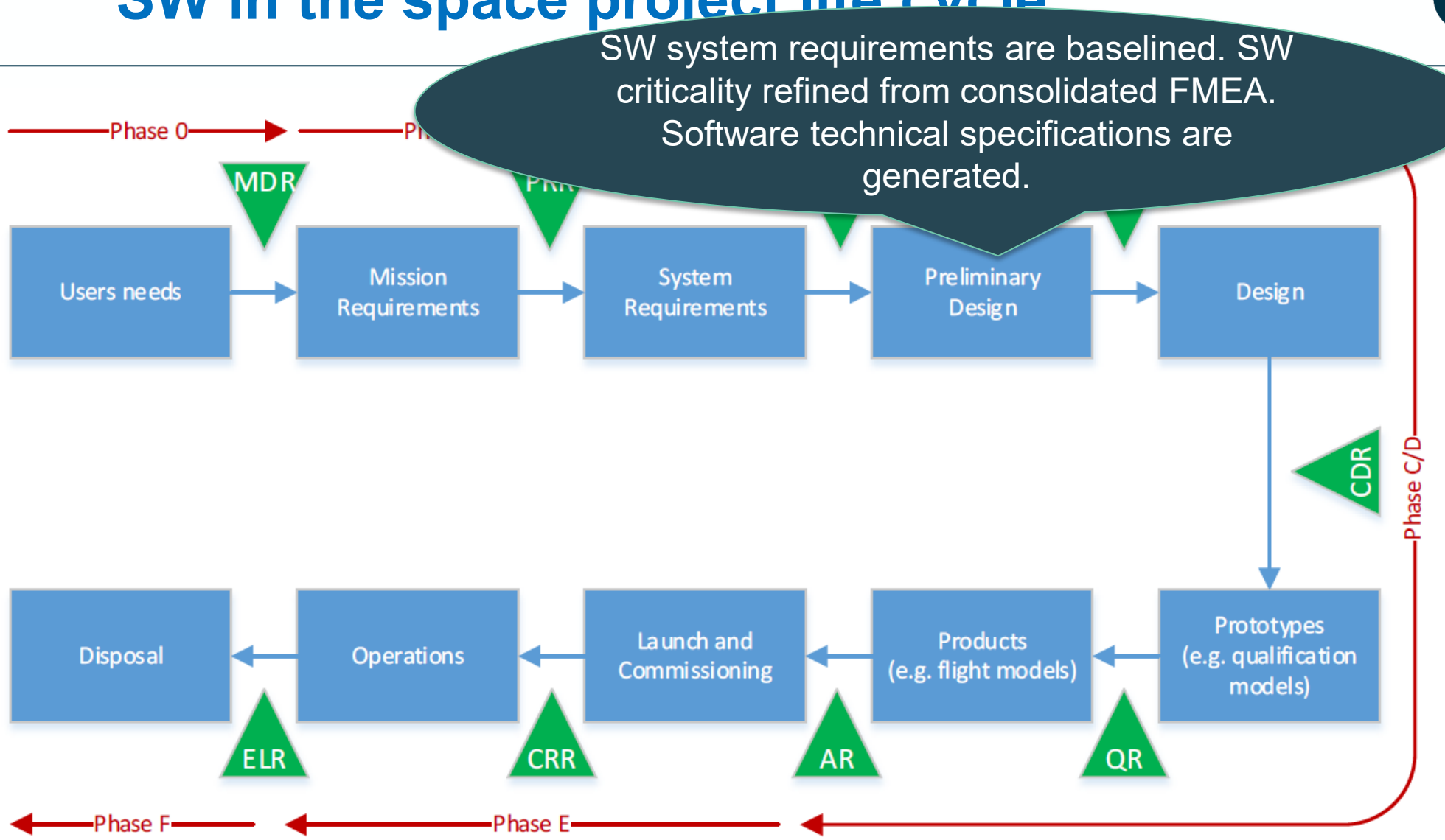
Project Phases 0, A and B

SW in the space project life cycle



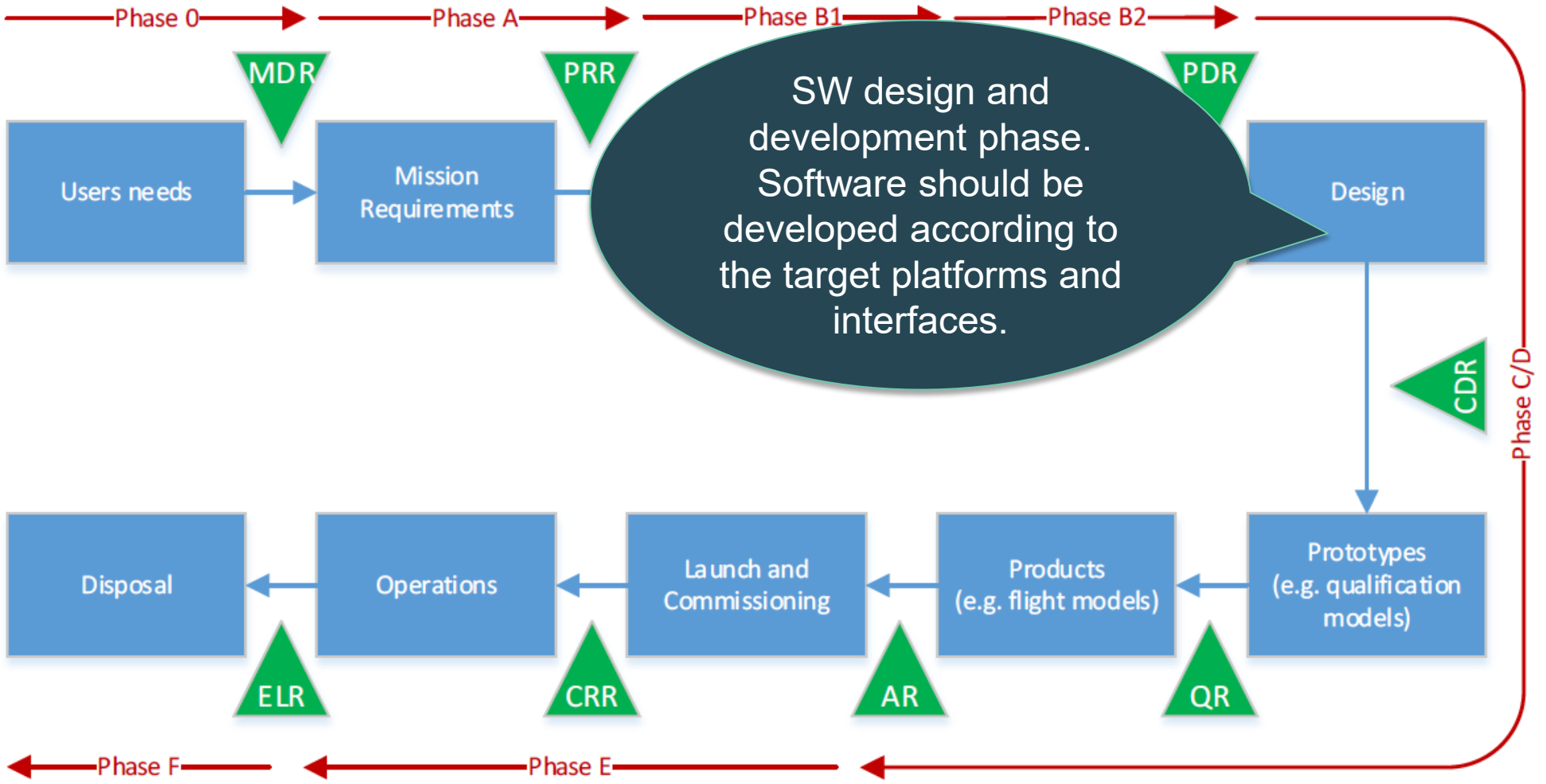
Project Phases 0, A and B

SW in the space project life cycle



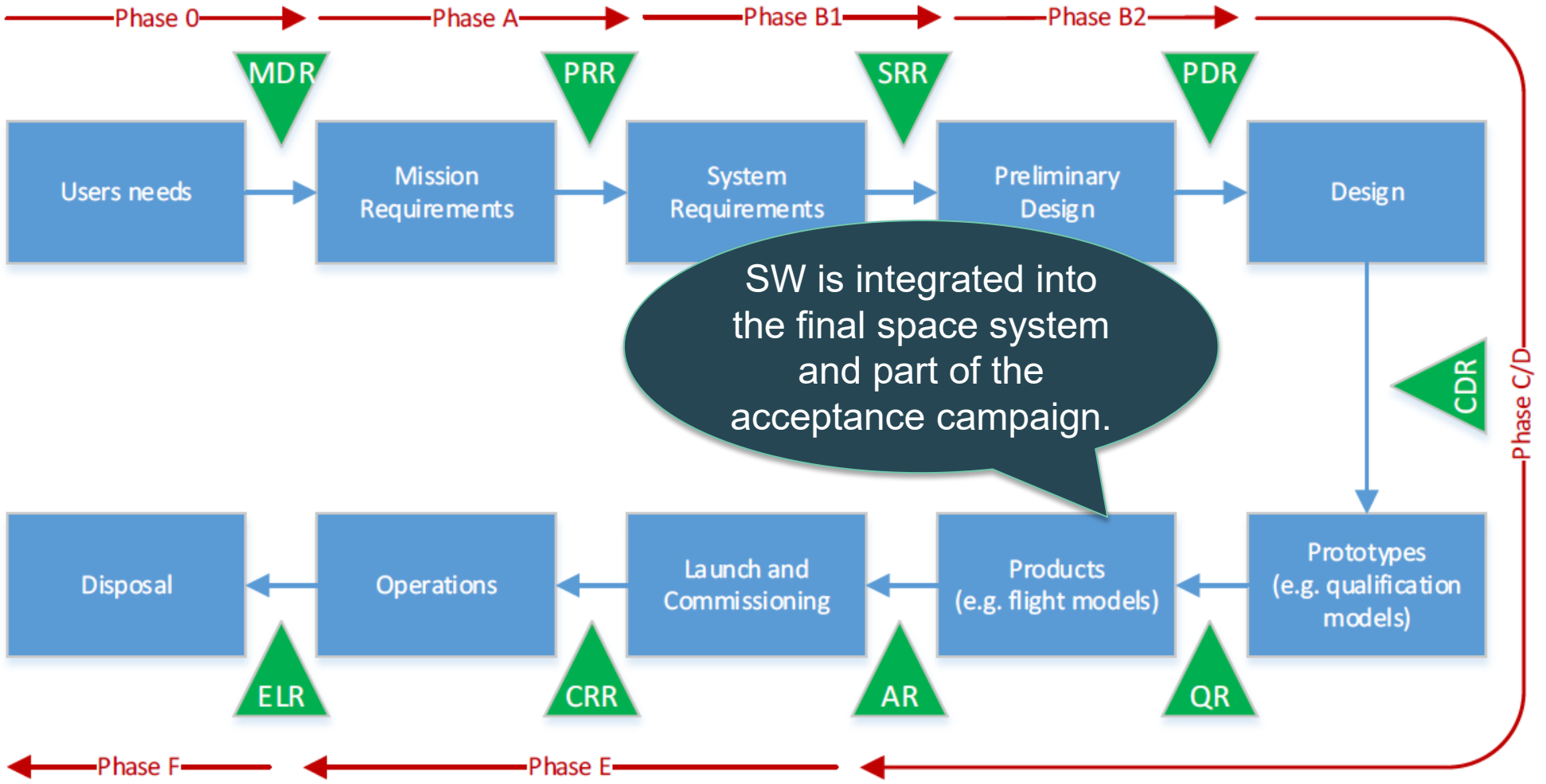
Project Phases 0, A and B

SW in the space project life cycle



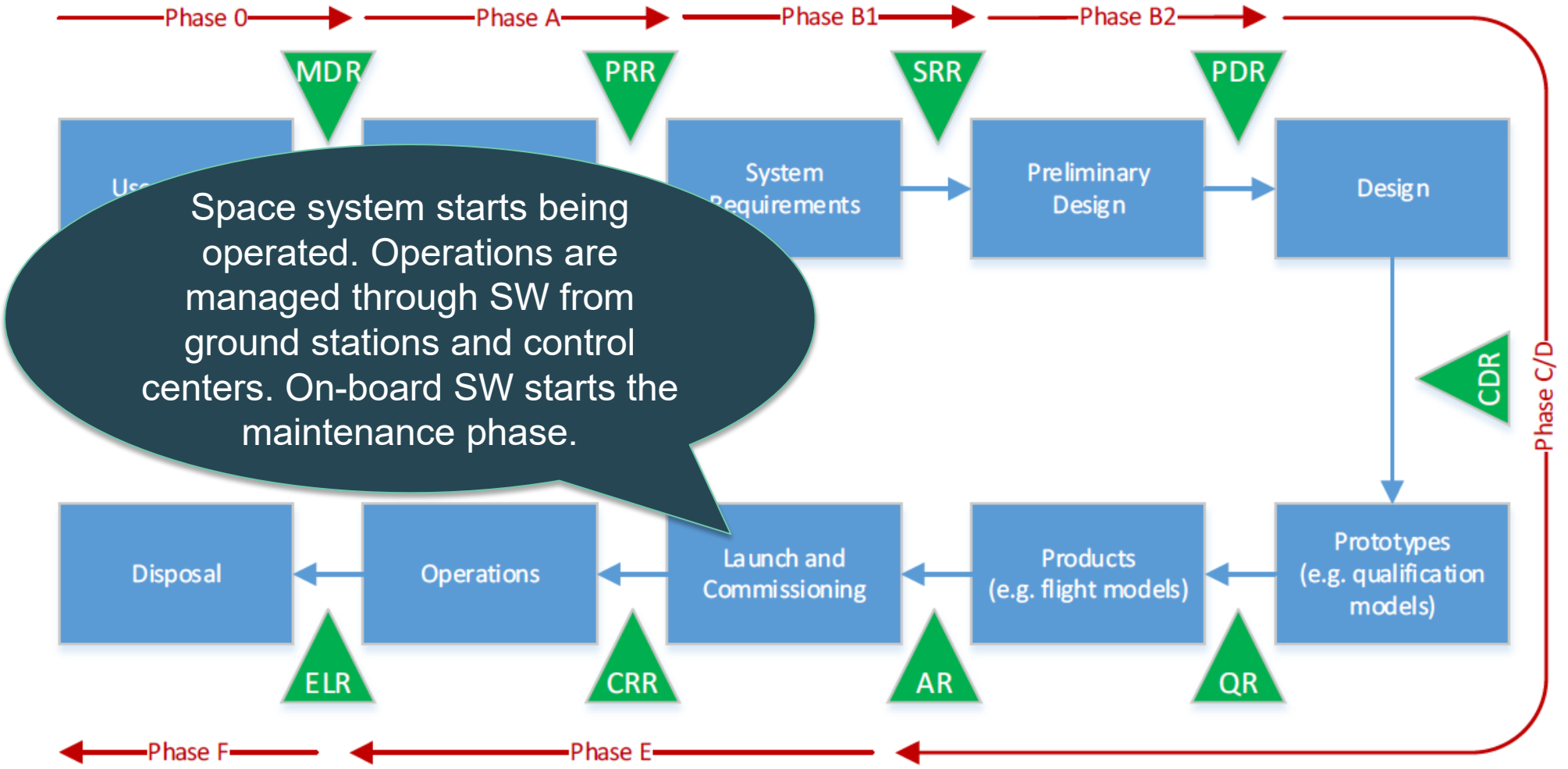
Project Phases C and D

SW in the space project life cycle



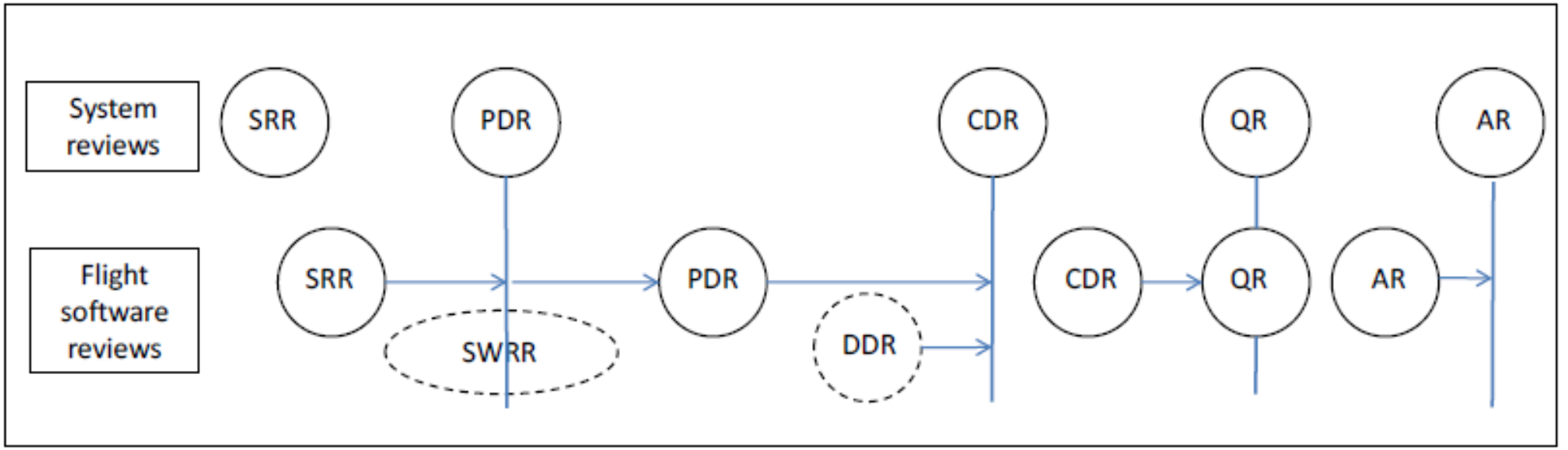
Project Phases C and D

SW in the space project life cycle



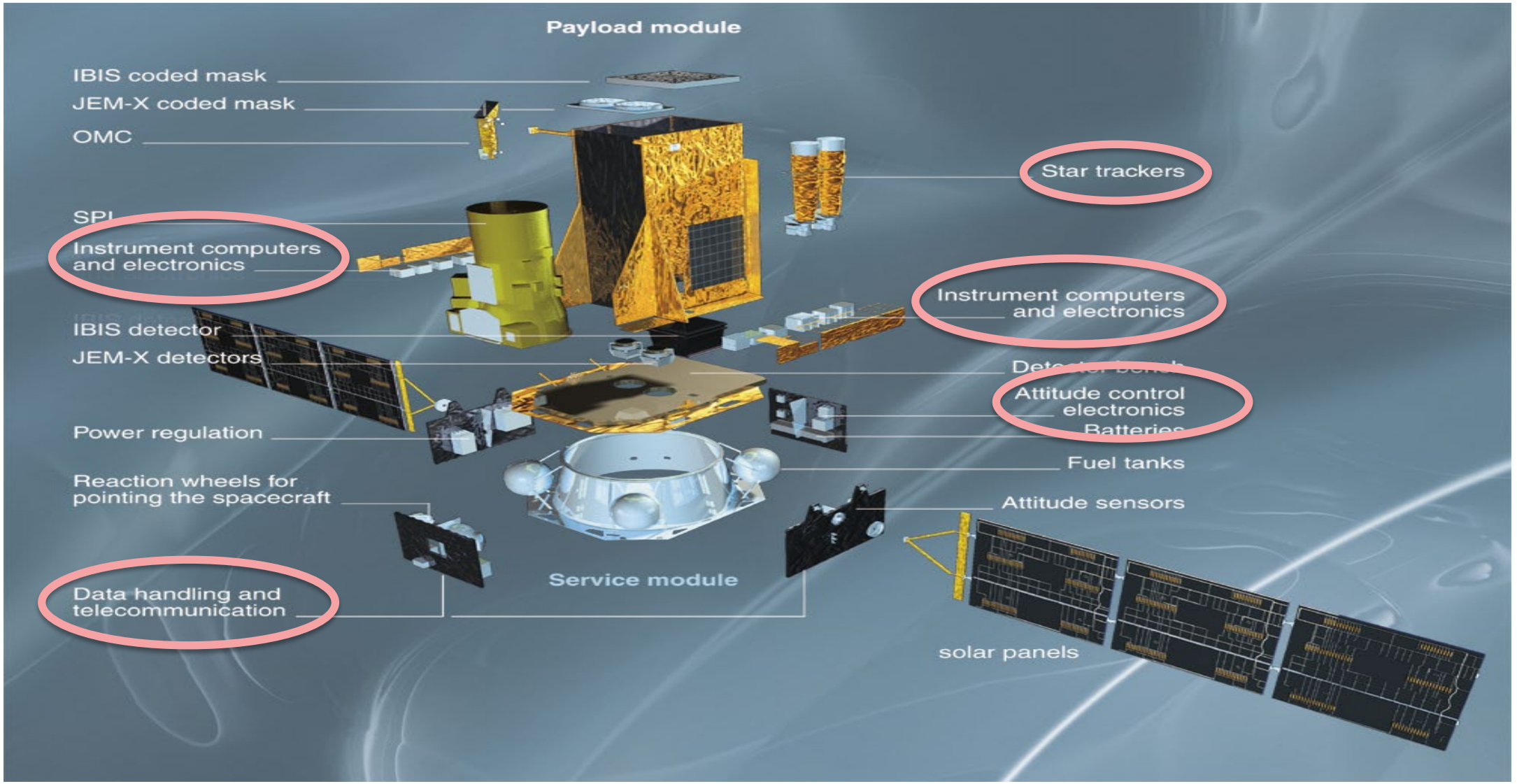
Project Phase E

SW in the space project life cycle



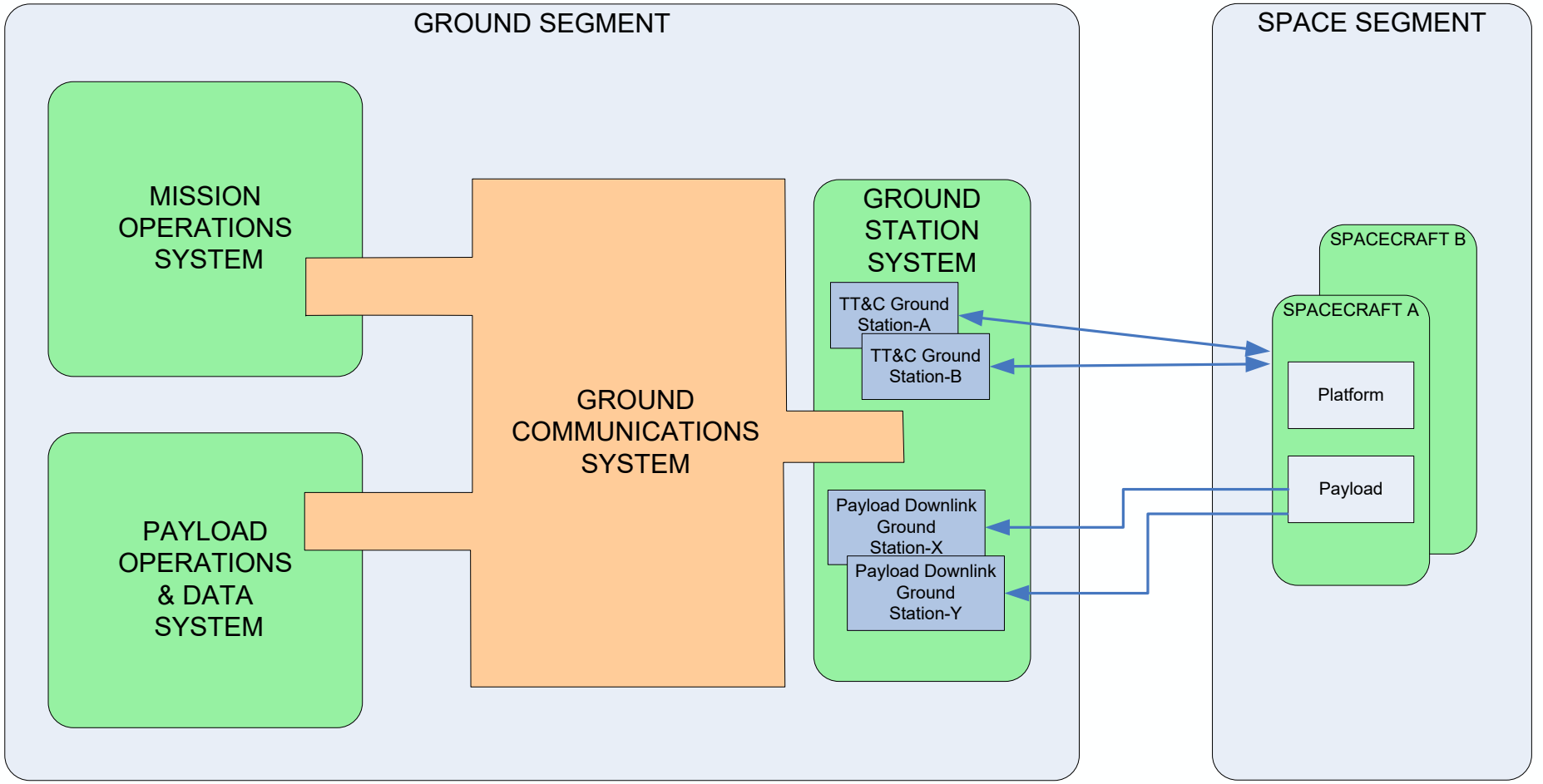
SW Reviews w.r.t System Reviews

Space software environments (I)

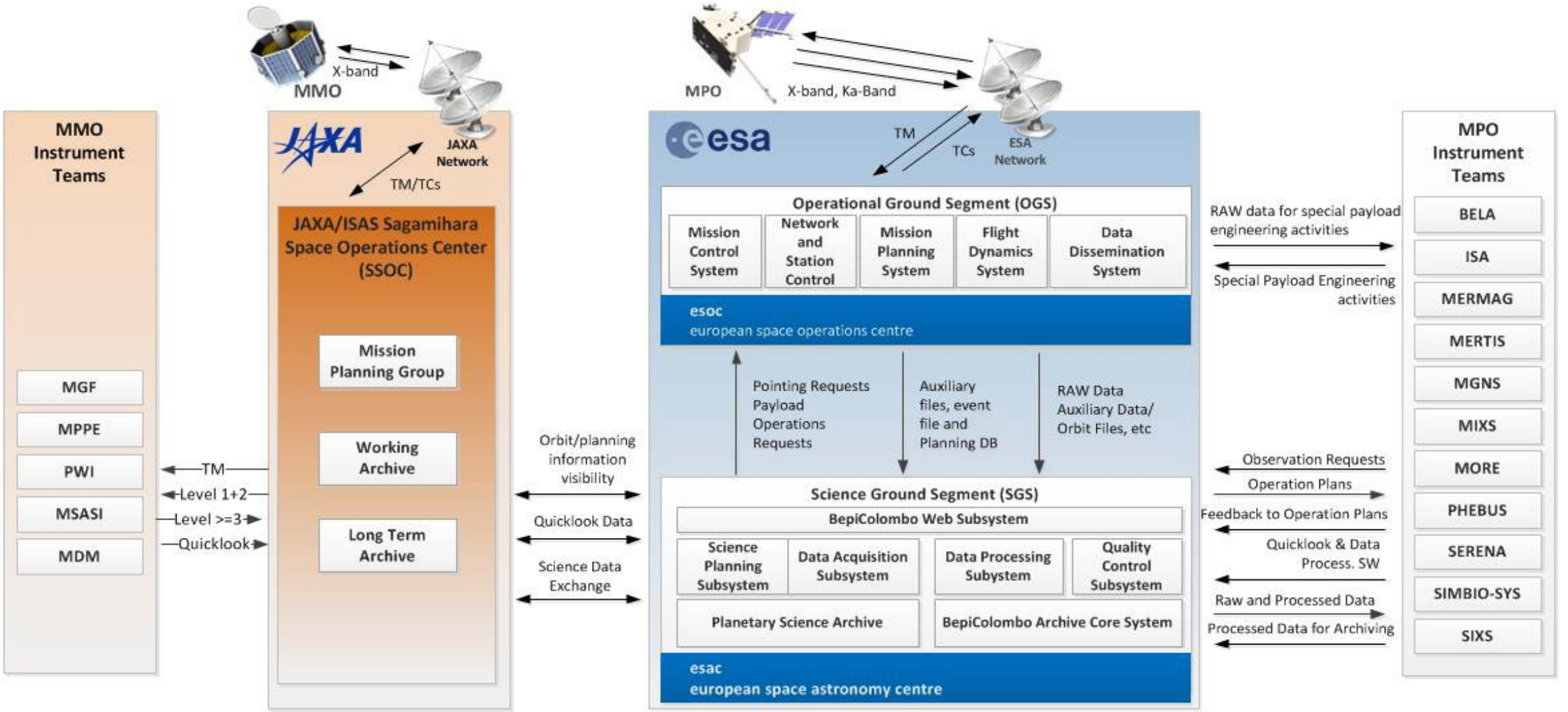


Space software environments (II)

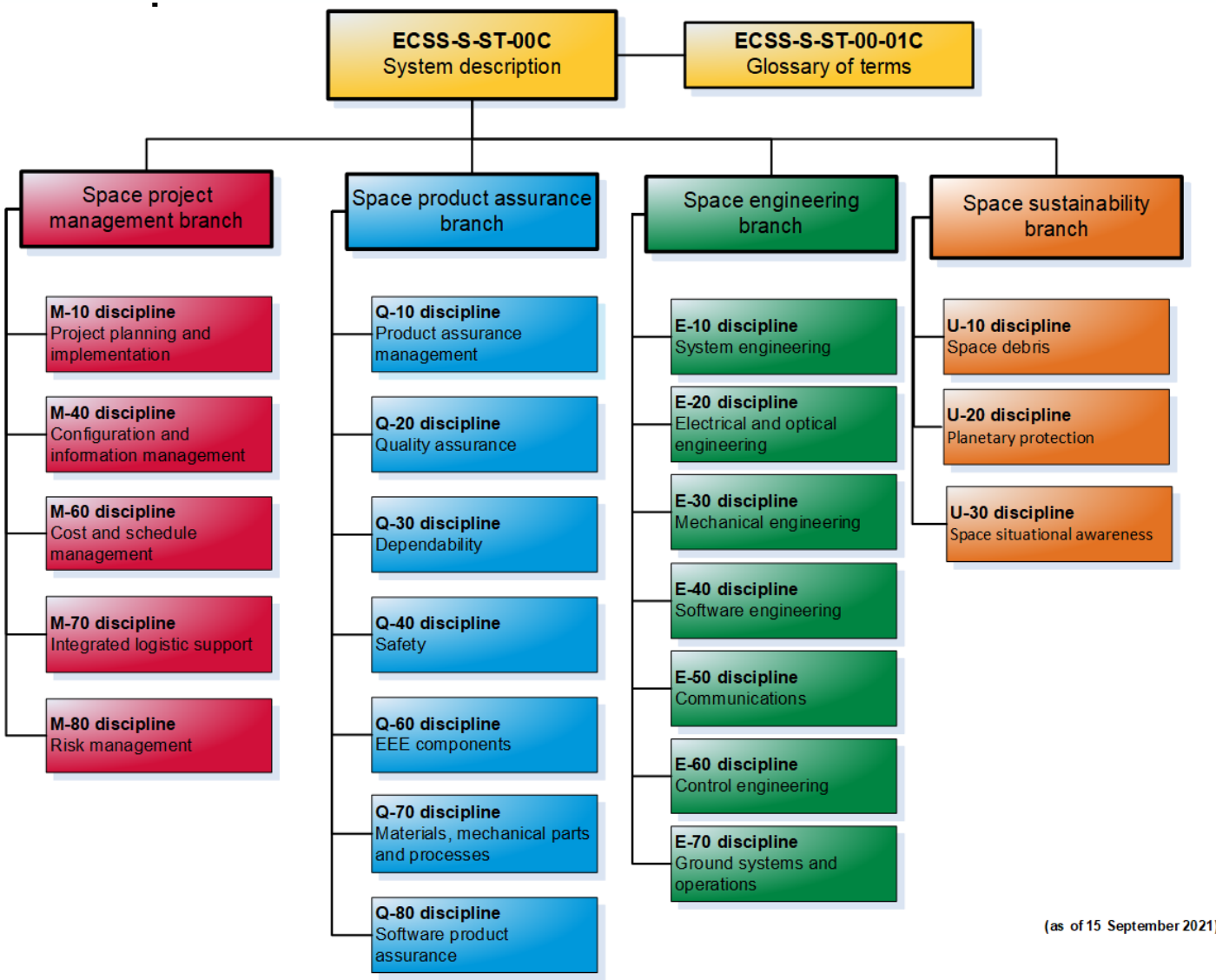
Ground and space segment



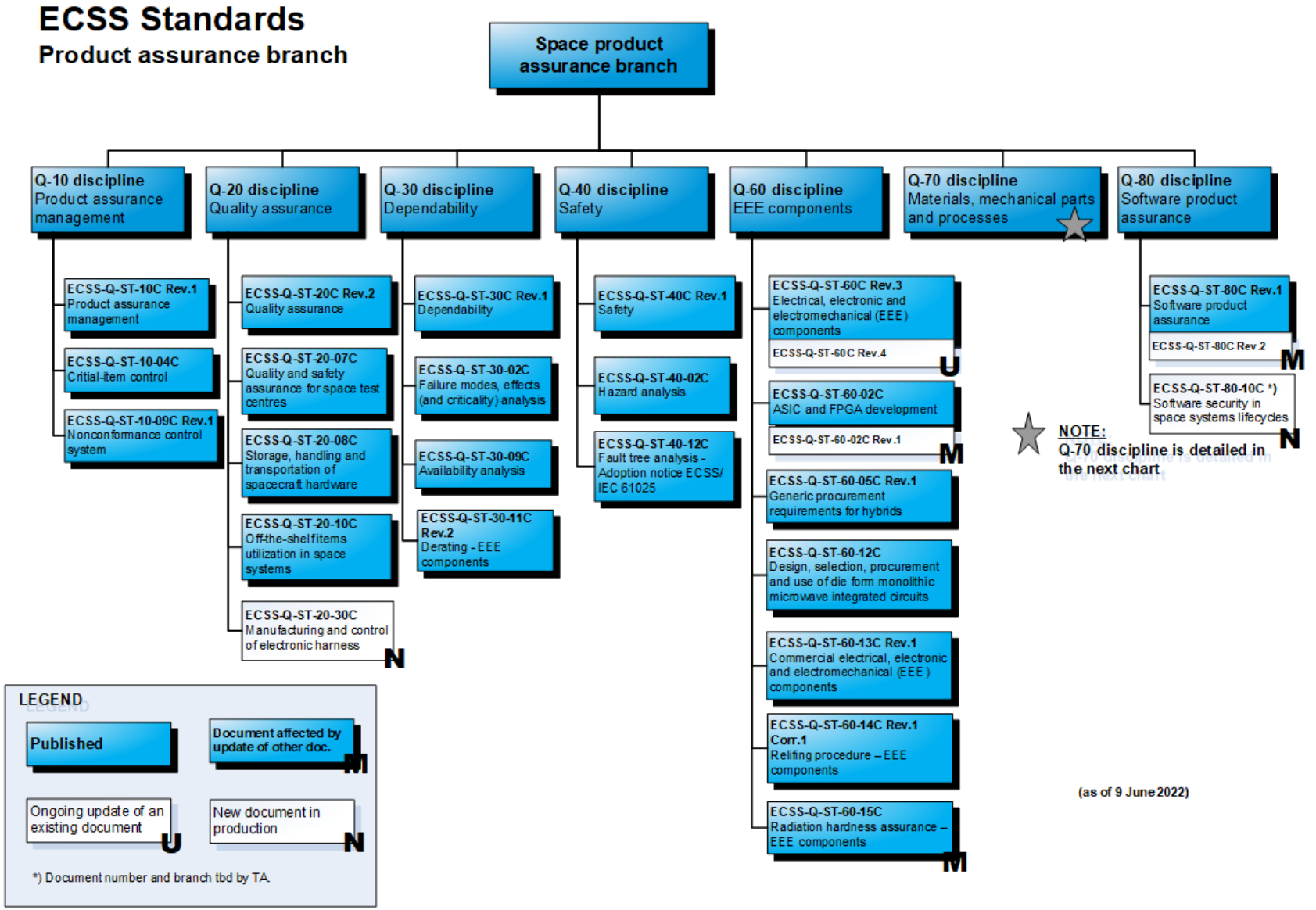
Bepi Colombo Ground Segment



ECSS Standards (ecss.nl)



ECSS Standards – PA branch



Relevant ECSS standards and handbooks (ecss.nl)

ECSS-E-ST-40C - Software

ECSS-Q-ST-80C Rev. 1– Software product assurance

ECSS-Q-ST-30C Rev. 1 – Dependability

ECSS-Q-ST-40C Rev. 1 - Safety

ECSS-E-ST-10C - System engineering general requirements

ECSS-M-ST-40C - Configuration and information management

ECSS-E-HB-40A – Software engineering handbook

ECSS-E-HB-40-01A - Agile software development handbook

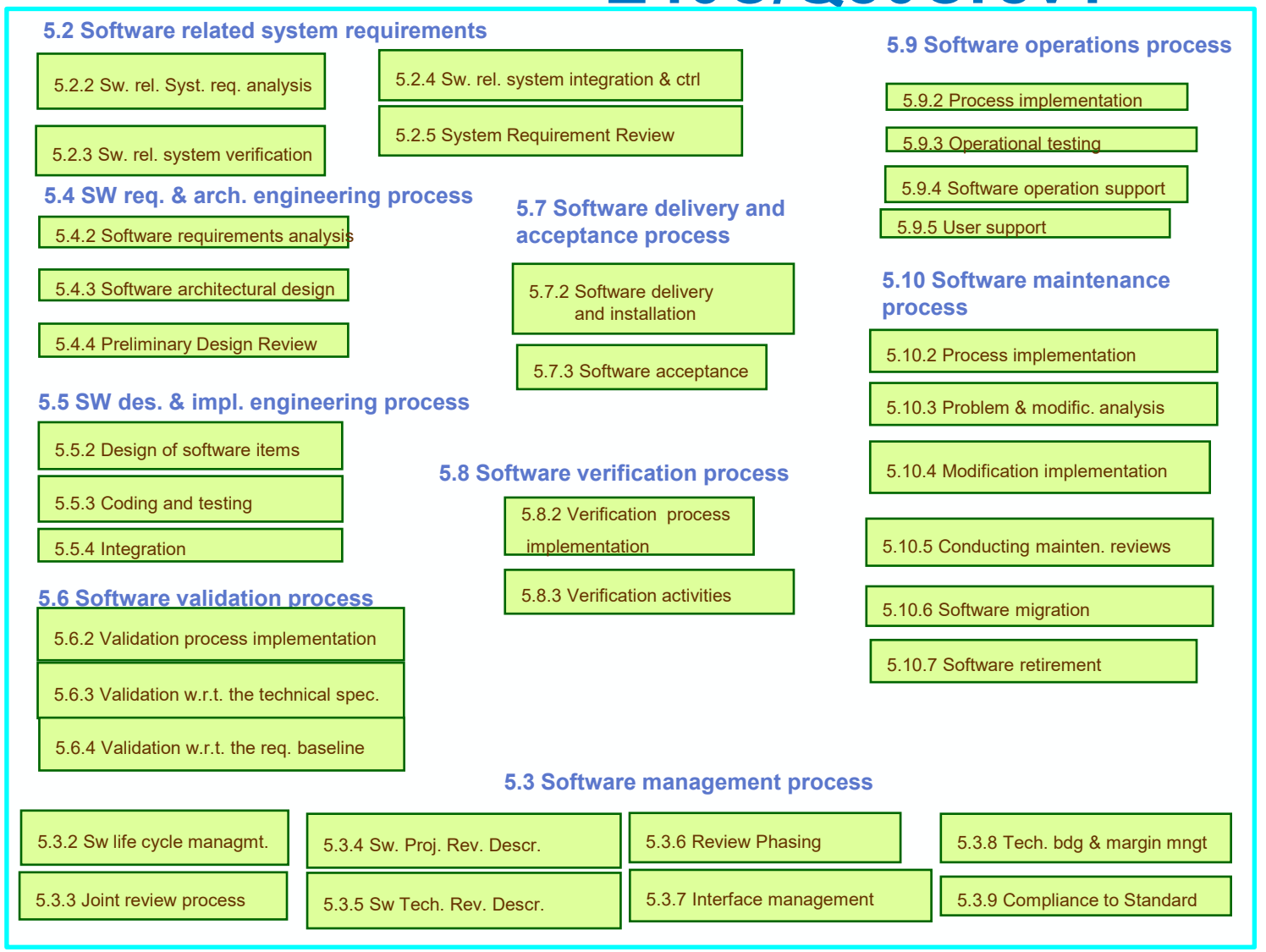
ECSS-Q-HB-80-01A – Reuse of existing software

ECSS-Q-HB-80-02 1A/2A - Software process assessment and improvement

ECSS-Q-HB-80-03A Rev.1 - Software dependability and safety

ECSS-Q-HB-80-04A - Software metrication programme definition and implementation

Software processes Requirements in ECSS E40C/Q80Crev1

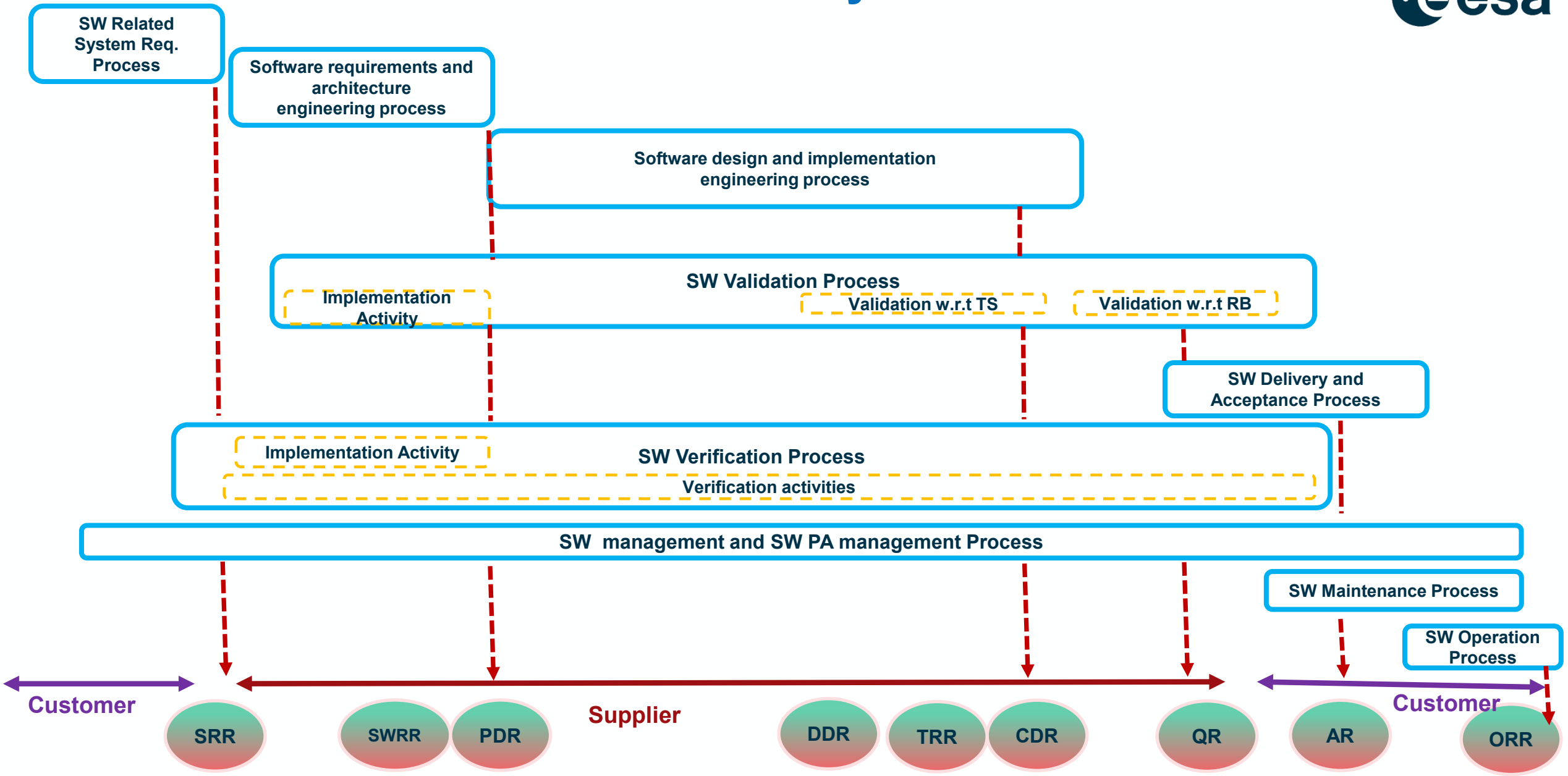


Software product assurance programme implementation	
5.1 Organization and responsibility	5.5 Procurement
5.2 Software product assurance programme management	5.6 Tools and supporting environment
5.3 Risk management and critical item control	5.7 Assessment and improvement process
5.4 Supplier selection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

Software life-cycle Process



Software project reviews

System requirement review (SRR)

After completion of the software requirements baseline specification, a software requirements review (SRR) shall take place.

Typical objectives:

- Agree with the customer or their representatives that all requirements captured in the requirements baseline are commonly understood and agreed.
- Review and baseline of the Requirements Baseline.
- Suitability of the draft software development plan including the software planning elements.
- Consistency of the software planning elements with respect to the upper level planning.
- Ensurance that software product assurance activities are performed.
- Evaluation of readiness to proceed to the next phase.

Software project reviews

Preliminary design review (PDR)

After completion of the software requirement analysis and architectural design, and the verification and validation processes implementation, a preliminary design review (PDR) shall take place.

Typical objectives (I):

- Agree with the customer or their representatives that all requirements with respect to the requirements baseline are captured in the technical specification.
- Review and baseline the Technical Specification.
- Review and baseline the software development approach and relevant plan.
- Review and baseline the software product assurance approach and relevant plan.
- Review and baseline the software configuration management approach and relevant plan.

Software project reviews

Preliminary design review (PDR) - cont

Typical objectives (II):

- Review and baseline the software verification and validation approach and relevant plan.
- Review and baseline of the software architecture.
- Review the technical budget and margins estimations.
- Review the integration strategy.
- Evaluation of the potential re-use of the software if applicable.
- Review of known unsolved issues which can have major impacts.
- Review the quality assurance reports.
- Evaluation of readiness to proceed to the next phase.

Software project reviews

Test readiness review (TRR)

A test readiness review (TRR) shall be held before the beginning of each significant test campaign, as defined in the software development plan.

Typical objectives:

- Baseline of the relevant test specifications (including test cases and procedures).
- Review of the testing facilities configuration.
- Verify that software documentation, software code, procured software and support software and facilities are under proper configuration control.
- Baseline the testing configuration.
- Review the quality assurance reports.
- Review the status of all open problems (SPRs, NCRs).
- Evaluation of readiness to proceed to testing.

Software project reviews

Critical design review (CDR)

After completion of the design of software items, coding and testing, integration and validation with respect to the technical specification, a critical design review (CDR) shall take place.

Typical objectives (I):

- Baseline of the detailed design (including the verification reports and technical budget report).
- Adequacy of the software units and integration plans and of the included unit and integration test procedures.
- Review and baseline the SValP approach and relevant plan.
- Review of the Software Reuse File, evaluation of the potential re-use of the software intended for reuse.
- Baseline of the validation specification w.r.t. the technical specification.
- Review of the unit and integration test results, including as-run procedures.

Software project reviews

Critical design review (CDR) - cont

Typical objectives (II):

- Verification that all the Technical Specification has been successfully validated (validation report) and verified (including technical budget, memory and CPU, and code coverage).
- Verify that the Software Configuration Item under review is a formal version under configuration control.
- Review of the software user manual.
- Review of known unresolved issues which can have major impact and resolution plan identification.
- Review the quality assurance reports.
- Review of the RB-validation facilities.
- Baseline of the Validation specification against the RB.
- Evaluation of readiness to proceed to the next phase.

Software project reviews

Qualification review (QR)

After completion of the software validation against the requirements baseline, and the corresponding verification activities, a qualification review (QR) shall take place.

Typical objectives (I):

- To verify that the software meets all of its specified requirements, and in particular that verification and validation process outputs enable transition to ”qualified state” for the software products.
- Review of the RB-validation test, analysis, inspection or review of design results, including as-run procedures.
- Verification that all the Requirements Baseline and interfaces requirements have been successfully validated and verified (including technical budgets and code coverage).

Software project reviews

Qualification review (QR) - cont

Typical objectives (II):

- Review the software release document.
- Review of the acceptance facilities configuration.
- Verify that the Software Configuration Item under review is a formal version under configuration control.
- Review of known unresolved issues which can have major impact and resolution plan identification.
- Review the quality assurance reports.
- Evaluation of readiness to proceed to the next phase.
- Review of the maintenance plan.
- Review the acceptance test plan.

Software project reviews

Acceptance review (AR)

After completion of the software delivery and installation, and software acceptance, an acceptance review (AR) shall take place.

Typical objectives (I):

- Review of the acceptance test results, including as-run procedures.
- Verify that the Software Configuration Item under review is a formal version under configuration control.
- Verification that all the RB software requirements have been successfully validated and verified (including technical budgets and code coverage) throughout the development life cycle.
- Baseline of the software acceptance data package.

Software project reviews

Acceptance review (AR) - cont

Typical objectives (II):

- Verify that the complete set of acceptance test cases is run on the same software version.
- Acceptance of the software product.
- Review of the software release document, the installation procedure and report and the maintenance plan.
- Review of known unresolved issues which can have major impact and identification of resolution plan for each outstanding issue and known problems.
- Correct closure of major SPRs/NCRs.
- Review of RFWs.
- Review the quality assurance reports.

ECSS-E-HB-40A Software life-cycle models

Model	Requirement Baseline	Iterative life-cycle	Short iteration	Risk driven	Utilization comment
Waterfall model	Frozen	No	N/A	No	When the customer needs are well-known and relatively stable. When no technological risks are involved.
Incremental model	Frozen	Yes	Depends	No	When the customer needs are well-known and relatively stable. When incremental deliveries are required. However, need to manage several versions in parallel. When no technological risks are involved.
Evolutionary model	Evolving	Yes	No	No	When, even with enough mature customer needs, the Requirements Baseline consolidation needs to be achieved in several steps. However, need a design robust enough to anticipate the RB evolutions. When early deliveries are required. However, need to manage several versions in parallel. When no technological risks are involved.
Spiral model	Evolving	Yes	Depends	Yes	When there are technological risks, e.g. not enough mature needs, new technology. However, require a larger schedule and cost to mitigate the risks.
Agile	Depends	Yes	Yes	Yes	When there is a need to quickly and continuously adapt and satisfy customer needs changes (e.g. on requirements or delivery needs). However, there is a risk that the final software does not meet the initial requirements and budget. For small software teams, when there is a strong and continuous involvement of the Customer throughout the development as well as co-locative customer/supplier teams.
Multi-level	Frozen	No	N/A	No	For very large software with complex industrial organisation.

Example of incremental model versions

Here is an example of versions definition and review setup in an evolutionary life cycle:

a. V1 is intended to allow the integration of all electrical units on the Electrical Functional Model (EFM). Therefore, V1 contains the basic Data Handling functions (typically a limited number of the functions). The architecture is defined, as well as some part of the detailed design. The technical budgets are still estimated. V1 must be tested enough such that it is usable on the hardware.

b. V2 is intended to allow the run of most close loop tests on the EFM using most of the AOCS involved equipment units. Therefore V2 contains the AOCS and Safe Mode (V2 is now typically a large number of the functions). The detailed design is more complete. Technical budgets are confirmed with high level of confidence. V2 is unit tested and validated on the stable functions.

c. V3 is intended to allow the satellite qualification. Therefore V3 contains the complete software. The technical budgets are measured and proven. V3 is fully verified and validated with 100% coverage of requirements.

d. V4 is the final flight software. V4 contains corrections of anomalies from system qualification. V4 undergoes regression tests and representative mission simulation.

NOTE The software located in non-erasable memory (PROM) should be ready at equipment CDR. Therefore its lifecycle should be shorter. The boot software should have at least passed a CDR for V1 delivery and should have its QR before the SW V2 TRR, or before the starting of the qualification of the first flight model.

	V1		V2		V3		V4	
	Project	Technical	Project	Technical	Project	Technical	Project	Technical
SRR	X		X		X			
PDR	X		X			X		
DDR		X	X			X		
TRR				X		X		
TRB/DRB		X		X				
CDR					X			
QR					X			
AR							X	

Disclaimer

DRD example: SPAP in Q80C rev1 Annex B

Annex B (normative)
Software product assurance plan (SPAP) - DRD

B.1 DRD identification

B.1.1 Requirement identification and source document

The software product assurance plan (SPAP) is called from the normative provisions summarized in Table B-1.

Table B-1: SPAP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses

ECSS Standard	Clause	DRD section
ECSS-Q-ST-80	5.1.2.1	<5.1>.a, <5.1>.b
	5.1.2.2	<5.1>.a, <5.1>.b
	5.1.2.3	<5.1>.b
	5.1.3.1	<5.3>
	5.1.4.1	<5.1>.b
	5.2.1.1	All
	5.2.1.3	All
	5.2.1.4	<5.10>
	5.2.1.5	<8>
	5.2.6.1c	<6.4>
	5.2.7.2	<5.5>
	5.4.3.3	All
	5.4.3.4	All
	5.6.1.1	<5.8>
	6.1.1	<6.1>
	6.1.5	<6.1>.c
	6.2.1.4	<6.2>
	6.2.3.2	<6.3>.c
	6.2.3.4	<6.3>.c
	6.2.3.5	<6.3>.c

ECSS Standard	Clause	DRD section
	6.2.4.8	<6.4>.d
	6.2.4.9	<6.4>.d
	6.2.4.11	<6.4>.d
	6.2.5.1	<6.5>.e
	6.2.5.2	<6.5>.e
	6.2.7.2	<6.6>.f

<6.2> **Projects plans**

a. The SPAP shall describe all plans to be produced and used in the project.

b. The relationship between the project plans and a timely planning for their preparation and update shall be described.

<6.3> **Software dependability and safety**

a. The SPAP shall contain a description and justification of the measures to be applied for the handling of critical software, including the analyses to be performed and the standards applicable for critical software.

<6.4> **Software documentation and configuration management**

a. The SPAP shall describe the contribution of the software product assurance function to the proper implementation of documentation and configuration management.

b. The nonconformance control system shall be described or referenced. The point in the software life cycle from which the nonconformance procedures apply shall be specified.

c. The SPAP shall identify method and tool to protect the supplied software, a checksum-type key calculation for the delivered operational software, and a labelling method for the delivered media.

<6.5> **Process metrics**

a. The SPAP shall describe the process metrics derived from the defined quality models, the means to collect, store and analyze them, and the way they are used to manage the development processes.

<6.6> **Reuse of software**

a. The SPAP shall describe the approach for the reuse of existing software, including delta qualification.

<6.7> **Product assurance planning for individual processes and activities**

a. The following processes and activities shall be covered, taking into account the project scope and life cycle:
   1. software requirements analysis;
   2. software architectural design and design of software items;
   3. coding;
   4. testing and validation (including regression testing);
   5. verification;
   6. software delivery and acceptance;
   7. operations and maintenance.

<6.8> **Procedures and standards**

a. The SPAP shall describe or list by reference all procedures and standards applicable to the development of the software in the project.

<8> **Compliance matrix to software product assurance requirements**

a. The SPAP shall include the compliance matrix to the applicable software product assurance requirements (e.g. ECSS-Q-ST-80 clauses, as tailored by a product assurance requirements document), or provide a reference to it.

b. For each software product assurance requirement, the following information shall be provided:
   1. requirement identifier;
   2. compliance
      (C = compliant, NC = non–compliant, NA = not applicable);
   3. reference to the project documentation covering the requirement (e.g. section of the software product assurance plan);
   4. remarks.

5.2.1.5

a. The supplier shall provide with the software product assurance plan a compliance matrix documenting conformance with the individual software product assurance requirements applicable for the project or business agreement.

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

b. For each software product assurance requirement, the compliance matrix shall provide a reference to the document where the expected output of that requirement is located.

NOTE For compliance with the required DRDs a general statement of compliance is acceptable.

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

Note: MLFS qualification Data pack can be downloaded from https://essr.esa.int/

DRD example: SDP in E40C Annex O

Annex O (normative)
Software development plan (SDP) - DRD

O.1 DRD identification

O.1.1 Requirement identification and source document

The software development plan (SDP) is called from the normative provisions summarized in Table O-1.

Table O-1: SDP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses

ECSS Standard	Clauses	DRD section
ECSS-E-ST-40	5.3.2.1a.	<5.2.1>
	5.3.2.1b.	<5.2.1>
	5.3.2.1c.	<5.1>, <5.3>, <5.4>
	5.3.2.1d.	<5.2.3>, <5.5>
	5.3.2.2	<5.2.1>
	5.3.2.3	<4.8>
	5.3.2.4a.	<5.2>
	5.3.2.4b.	<5.3>
	5.3.2.4c.	<5.3>
	5.3.2.4d.	<5.4>
	5.3.3.2a.	<5.2.3>
	5.3.3.3a.	<5.2.3>
	5.3.3.3b.	<5.2.3>
	5.3.3.3c.	<5.2.3>
	5.3.6.1a.	<5.2.2>
	5.3.6.1b.	<5.2.2>
	5.3.6.2	<5.2.2>
	5.3.9.1	<5.6>
	5.3.9.2	<5.6>
ECSS-Q-ST-80	5.6.2	<4.8>
	5.7.2.1	<5.4>
	5.7.2.2	<5.4>
	6.3.4.5	<5.4>a.

160

ECSS

ECSS-E-ST-40C
6 March 2009

<5> Software development approach

<5.1> Strategy to the software development

a. The SDP shall describe the overall strategy to the software development.

NOTE An activity diagram can be included.

<5.2> Software project development life cycle

<5.2.1> Software development life cycle identification

a. The SDP shall describe the software development life cycle.

b. Definition of the selected life cycle paradigm (e.g. waterfall, incremental, or evolutionary) as well as the adopted software versioning approach shall be included.

c. The SDP shall cover the implementation of all the activities and tasks relevant to the involved software processes, including:

— system engineering processes related to software;

— software requirement & architecture engineering process;

— software design and implementation engineering process;

— software validation process;

— software verification process;

— software delivery and acceptance;

— software operation process;

— software maintenance process and its interface with development (documents to be handed over, tools to be maintained);

— software management process.

<5.2.2> Relationship with the system development cycle

a. The SDP shall describe the phasing of the software life cycle to the system development life cycle.

NOTE A process model representation can be used.

<5.2.3> Reviews and milestones identification and associated documentation

a. The SDP shall describe scope and purpose of each identified review, relevant deliverable and expected outputs.

163

this clause

<5.6> This Standard's tailoring traceability

a. The SDP shall include the coverage matrix of the applicable tailoring of ECSS-E-ST-40 clause 5, or provide a reference to it.

5.3.9 Compliance to this Standard

5.3.9.1 Compliance matrix

a. The supplier shall provide a compliance matrix documenting conformance with the individual software engineering process requirements (Clause 5) applicable to the project or business agreement, as per ECSS-S-ST-00.

EXPECTED OUTPUT: ECSS-E-ST-40 compliance matrix [MGT, SDP; SRR, PDR]

5.3.9.2 Documentation compliance

a. The compliance to each of the individual software engineering process requirements shall make reference to the document where the expected output is placed, if it is not placed in this Standard's DRDs in annexes of this document.

EXPECTED OUTPUT: ECSS-E-ST-40 compliance matrix [MGT, SDP; SRR, PDR]

NOTE A general statement of compliance to this Standard's DRDs is acceptable.