

Guidelines for the tailoring of ECSS-E40

1 Introduction

This document provides guidelines to support the tailoring of the ECSS-E40. It is intended for technical officers at Estec who have written a statement of work for an activity that includes a software development. The goals of the tailoring are:

- To help the technical officer to have a complete view of the software project from his management standpoint and to focus the software effort according to the project context,
- To verify the completeness of the statement of work for the software,
- To clarify the software activities to the bidders in order to help them to prepare better proposals,
- To help TEB members to evaluate the proposals from a software standpoint by checking the bidders' detailed compliance to software requirements,
- To improve the achievements of the software projects and contribute to the reduction of the "software crisis"...

ECSS-E40B covers all possible software developments, organized in processes:

- System Requirement Engineering related to Software,
- Software Management
- Software Requirement & Architecture Engineering
- Software Design & Implementation Engineering
- Software Validation
- Software Delivery and Acceptance
- Software Verification
- Software Operations
- Software Maintenance

Each process includes a set of requirements to be complied with by the bidder to the statement of work. The tailoring is the selection or not of each of these requirements. It must be done by the Customer (i.e. you). It is performed in three steps:

- The project characterisation (role/importance of the software in the system, estimation of the risky area, roles)
- The tailoring table indicating the selected requirements of the original E40. The reply to each question of the questionnaire allows to select or not a set of requirements. The selected requirements can be reported in the tailoring table of the tailoring annex of the statement of work..
- The justification for the tailoring and the mapping to your statement of work.

For any questions or assistance in the tailoring, please contact TEC/EME. The availability of an automatic tool to support the tailoring is planned.

2 Your statement of work

Your statement of work must include:

- In the Applicable Document list, the document ECSS-E40 Part1, Version B, 28 November 2003 and ECSS-E-40 Part2 Version B
- A tailoring annex based on the *E40B tailoring annex template*, which describes the tailoring. The result of the questionnaire (included in these guidelines) is used to fill in the *tailoring table* of the tailoring annex template.

The tailoring annex is designed to provide the reader with information in a logical order for him. However, it is more convenient for you to fill it in a slightly different order.

3 The tailoring annex

1→ FILL IN [ROLE/IMPORTANCE OF SOFTWARE IN THE SYSTEM]

The **role/importance** of the software in the system allows to position the software project within the whole activity, and to decide *the extent to which its contribution to the ultimate goal of the contract is essential*. It must also describe the various *software components*. Each of them will be subject to a tailoring. Practically, each of them is a column in the tailoring table.

A *component* is this part of your software development that has homogeneous characteristics. It can be an executable or a main function of the software. Examples of such components are an algorithm to be plugged into a tool, a man machine interface, a driver, a prototype, etc. The components are generally already described in the statement of work, but formally mentioned here.

In order to help you to characterize your project, here are some criteria/factors:

Technical factors	
Novelty of the domain of application	
Complexity of the software, as measured by the number of interface or similar metrics	
Amount of software that has to be produced	
Reusability required of the software being developed	
Interface to system development projects	
Degree of use of COTS or existing software	
Maturity of the COTS	
Completeness or stability of the user requirements	
Operational factors	
Type of application (platform, payload, experiment)	
Number of potential users of the software	
Criticality of the software as measured by the consequences of its	

failure	
Expected lifetime of the software	
Number of sites where the software is used	
Maintenance constraints	
Management factors	
Amount of effort required to develop the software	
Amount of time required to develop the software	
Budget requirements for implementing and operating the software	
Schedule requirements for delivering the software	
Number of people required to develop, operate and maintain the software	
Experience of the supplier	

2→ FILL IN [ESTIMATION OF THE RISKY AREAS]

The **estimation of the risky areas** allows you to estimate in which part of the software development the effort has to be placed. The above factors can be used to quantify the project and realise where the risk is:

- *Complex specification*: risk of bugs during validation, effort in software requirements
- *Tricky design*: risk of bugs during integration, effort in development
- *Need for reliability*: risk of user dissatisfaction, effort in validation,
- *Criticality*: risk of mission lost, effort in verification
- *Long term use*: risk in availability, effort on maintenance
- *Non trusted/unknown Supplier, interface with other projects, complex organisation*: risk in schedule, development plan and interface management, effort on project management

NOTE: Each time you tailor out an E40 requirement, you may increase a risk in your software project.

3→ FILL IN THE ROLES

The standard assign processes to **roles** (customer, supplier, etc). It is important to know who is doing what in your particular project.

4 → FILL IN THE QUESTIONNAIRE

The questions will now help you to get a more detailed view of your complete project. You will select which processes need to be involved, which information needs to be provided, which reviews need to be organized.

You can now go back to the tailoring annex.

5 → FILL IN THE TAILORING TABLE

Report the results of the questionnaire in the table (e.g. by writing YES in the selected requirements). If you have several components, foresee one column per components if they have a different tailoring. For example, you might have a component algorithm that you specify and design, another that you only design, and a complete tool that you only integrate and validate.

6 → FILL IN [JUSTIFICATION FOR THE MAIN TAILORING]

As a consequence of both the importance of the software and the risk assessment, completed by the context of your project given by the questionnaire (you have already system specification or not, you have that budget, your software has that technical characteristics, you need visibility on the detailed design or not, you need maintenance or not, etc) you may tailor out or merge some processes of the E40B process model, you may remove or merge some reviews, you may require or not visibility on the development. This is the **justification for the main tailoring**. This will allow the bidders to understand your views and to make a better proposal (they will also understand that they cannot cheat you in the domain of software engineering).

7 → MODIFY THE SECTION “PROCESSES INVOLVED”

As a result of the questionnaire, you have selected software engineering **processes**, and excluded others. Summarize this here, by moving the bullets in the right section.

8 → MAP THE SOFTWARE ENGINEERING TO THE STATEMENT OF WORK

The organisation in work packages of the statement of work is generally made from an application domain standpoint. The software activities, as they now appear from the tailoring, are likely to be spread over the work packages. The software reviews are likely to be hidden within system reviews or work package acceptance. The software work outputs are likely to be part of broader documents.

There are two goals here:

- **Clarify** for the software team of the bidder for which work package he has to participate in the proposal, he has to contribute in the schedule, he has to evaluate a cost,
- **Verify** that all your expected software activities are actually mentioned in your work package organisation.

The table lists in the left column the software processes and the software reviews. Just trace in the other columns (one per component) in which work package you expect the processes will be performed, and what are the names of the software reviews in the statement of work.

Revisit your statement of work in case something is missing.

9 → FILL IN THE DOCUMENTATION TABLE

The ECSS-E40 comes with a set of DRDs, describing the most complex software documents. By an ECSS rule, the DRDs are applicable. However, by mean of the tailoring, you may leave to your Supplier the flexibility to deliver its own documentation.

Some documents are mandatory, because some requirements of some processes cannot be tailored out.

This more exhaustive list of documents can be used to ask specific documents to your Supplier:

Document item	Acronym in DRD	Folder	Applicability
(Software) System Specification	SSS	RB	
Software Interface Requirements Document	-		
Software Requirements Specification	SRS	TS	
Software Interface Control Document	-		
Software Design Document (including Software Components Design)	SDD	DDF	
Software Source Code	-		
Software Configuration File	SCF		
Software Release Document	SRD		
Training material			
Software User Manual	SUM		
Software Reuse File	SRF	DJF	
Software Verification Plan	SverP		
Software Validation Plan	SValP		
ISVV Plan	-		
Software Units/Integration Test Plan	SUITP		
Software Validation Testing Specification wrt TS	SVTS		
(Analyses & Inspection) verification report wrt TS	-		
Software Validation Testing Specification wrt RB	SVTS		
(Analyses & Inspection) verification report wrt RB	-		
Software Traceability Matrices	-		
Software Acceptance Test Plan			
Software Requirements Verification Report	-		
Software Arch. Design and Interface Verification Report	-		
Software Detailed Design verification Report	-		
Software Code Verification Report	-		
Software Documentation Verification Report	-		
Software Integration Verification Report	-		
Software Unit/Integration Test Report	-		
Software Validation Test Report wrt TS	-		
Software Validation Test Report wrt RB	-		
Validation Evaluation Report wrt TS	-		
Validation Evaluation Report wrt RB			
Software Design & Test Evaluation Report	-		
Acceptance Test Report			
Installation Plan			
Installation Report			
Software Budget Report	-		
Software Acceptance Data Package	-		
Schedulability Analyses	-		

Software Behavior Verification	-		
Testing Feasibility Report	-		
Problems and Nonconformance Report	-		
Milestones Report	-		
Software Maintenance Plan	-	MF	
PR & NCR - Modification analysis report -Problem analysis report			
Migration Plan	-		
Software Operational Plan	-	OP	
Software Development Plan	SDP		
Software Product Assurance Plan	SPAP		
Software Criticality Analysis	-	PAF	
Software Product Assurance Report	-		

10 → FILL IN THE INTELLECTUAL PROPERTY RIGHTS

Although the Contract Officer will probably suggest you to move this part into the contractual documents, it is important that you prepare the future of your software. Would you like in the future to give access to the software to different companies, or instead to preserve a market by keeping a single proprietary source, or to explore the dynamism of open source development, then select your preferred **IPR scheme**, in agreement with the Contract officer.

1. The General Case

The general case is that the contractor owns full intellectual property rights with respect to computer software developed under contract to the Agency. However, the Agency and the Member States are entitled to a free-of-charge, non-exclusive, irrevocable license to use, reproduce and modify the software for their own requirements. This also includes the right to grant sub-licenses to third parties within the Member States for peaceful purposes in the fields of space research and technology, and space applications.

2. Operational Software

At the stage of invitation to tender, the Agency may request the ownership of the intellectual property rights of operational software, on conditions to be defined in the contract. Operational software is defined as "software required for essential space segment or check-out purposes and which has been developed with a fundamental intellectual contribution from the Agency's staff".

Operational software includes any software that is used in operations, either as part of the space segment or ground segment. Operational software is frequently

interpreted in the broader sense of any software used in the Agency's operations – this could include software supporting the Agency's business processes as well as technical software. The phrase “developed with a fundamental intellectual contribution from the Agency's staff” is taken to mean that the Agency's staff have either written or been deeply involved in the writing of the software requirements or in preparing any other major outputs of the software development (e.g. architectural design, coding).

This case gives the Agency full rights on the software's IPR in order to ensure its freedom to apply it and modify it as necessary for its operations. *In practice, ESA initiating authorities are recommended to use the operational software designation in all cases where there is an expectation or possibility that the software being procured may become operational (this may include pre-developments and technological activities).*

3. Co-funded Developments

In co-funded developments, both the Agency and the Contractor provide funding for the development of the software. Such arrangements are also referred to as “Public-Private Partnerships” (PPP). This approach has been used in certain of the Agency's technology programmes. Such co-funded developments should result in a product that the contractor can market commercially. Fundamentally, the intellectual property rights position is as for the general case. However, in order to protect his investment, the contractor may wish to restrict the license conditions further; for example, access to source code may be restricted or the rights to give sub-licenses may be restricted.

4. Public domain and open source

When the software is public domain, the IPRs belong to nobody. When the software is open source, the IPRs generally belong to the author, but there is an obligation to follow a pre-defined license that will propagate the open source characteristic of the software all along its development.

Note that software reusing a component licensed under the GNU license receives automatically the GNU license and becomes open source. Whereas software *linking* with a component licensed under the GNU *Library* license may not be open source. This is important for example when on-board software embed a GNU run-time system.

4 Questionnaire

The questionnaire is organized by process, with a specific section about questions impacting several processes. As a Technical Officer of the activity, you are the Customer. The bidder to your activity is the Supplier.

Note that there are mandatory processes including mandatory requirements, which you cannot tailor out without degrading your professionalism reputation to an unacceptable level. This minimum set of requirements is called "*ECSS-E40 for dummies*".

System Requirement Engineering Process (5.2)

The **System Requirement Engineering** process is a **System** process (common with ECSS-E10 Space Engineering, System) and also a **Customer** process. Therefore, it is **your** process. The software development, subject of the statement of work you are working on, is (part of) a system. The purpose of this process is to generate your requirements for your System as a System engineer or as a Customer. This is called the *Requirement Baseline*. There are three possibilities:

1) You need System Requirements; you have done them.

If you believe that your Requirement Baseline is complete, consistent, feasible (even if limited to a couple of pages as annex 1 of the statement of work), if you don't need the Supplier to consolidate it or to discuss it, if you don't need a System Requirement Review, then skip this process that is tailored out, and go to the next process.

2) You need System Requirements; you did not and will not do them.

If you don't have a solid Requirement Baseline, but you feel the need to get one separate from the Supplier's software specification, or in case your system includes specific hardware, you can **delegate** the elaboration of the Requirement Baseline to the Supplier, but keep the responsibility and the approval of it through the System Requirement Review. In this case, proceed with this process 5.2, where the Customer requirements are in fact for the Supplier to do, but under your responsibility.

3) You don't need specific requirements at system level; you think that they are the same as the software requirements.

If on the opposite you don't have a Requirement Baseline, but this is on purpose because you don't believe that it is useful, as you have very little to impose (no development constraints), as the Supplier knows the domain much better and you rely on him for the requirements, as your system is software intensive and not embedded into specific hardware (your software is the root in your product tree), then consider merging the System Requirement engineering and the Software Requirement & Architecture Engineering in a single process, called Software Requirement & Architecture engineering. This process is tailored out. Go to the next process.

NOTE: The quality of the requirements in general, and the Requirement Baseline in particular, has proven to **impact directly on the success** of the software project. The tailoring of this process, which in a way is a self-assessment of the customer, must be carefully performed. In case of doubt, *select the second option*; you will still have the SRR to include missing system requirements. The SRR also allows the Supplier to verify the Requirement Baseline, to adopt it and to suggest improvement. If you don't give your requirements, you will not be able to ask for them later in the project, or it will cost you a CCN with cost and delay.

5.2 System requirement engineering processes related to software

If you have selected this process, there are requirements that are always applicable (system requirements specification, system architecture, interface requirements, identification of development constraints, System Requirement Review [SRR]). Always select:	5.2.2.1	5.2.3.1	5.2.4.5	5.2.5.2	5.2.5.5	5.2.6.2	5.2.7.1	5.3.3.2 5.3.3.3	5.3.2.6 (If 5.3 is selected)
If a higher-level criticality analysis has demonstrated that your system is critical, or if a software failure impacts drastically on human life, mission goals, investments realised, then it is important to identify the software components/function that impact on the system/mission/investment or safety. Select:	5.2.2.2								
If your system includes software and hardware and/or human operations, then a system partitioning is necessary, select:	5.2.3.2								
If your system is so complex that it's future system and mission level validation deserves specific requirements to be expressed now (e.g. testing scenario, data, environment), then select:	5.2.4.2	5.2.4.3							
If the system is complex enough and involve specific hardware, you may want to produce additional requirements for the future system integration (system observability, data medium, customer's input, supplier's output, supplier's support), and for the future system operation (operational plan). Select:	5.2.5.1	5.2.5.3	5.2.5.7	5.2.5.8	5.2.5.9	5.2.6.1			
If the software includes a Man Machine Interface complex enough to need ergonomics guidelines and mock up, select:	5.2.2.3	5.2.2.4	5.4.2.6						
If the software is developed in order to be reused in other projects, select:	5.2.5.6	5.4.3.9	5.4.3.10						
If the software development activities aim to deliver flight software	5.2.5.1	5.2.5.4							

Software Management Process (5.3)

The **Software Management** process is a process of the **Supplier**. The software project must be handled as another project; therefore most of the requirements have to be found in the ECSS M branch. However, there are specific information that the Supplier must provide for software such as the *software life cycle*, the *software development environment* and the *technical budgets*.

If you believe that the project follows a classical life cycle and uses classical methods and tools well known by you and your supplier, where the consequences of having a poor management would not increase the risks to an unacceptable level, and where the technical budgets are not important to monitor, then you may skip this process. Instead, you must ask your Supplier for a draft Software Development Plan (including life cycle, methods, tools and technical budgets) in its proposal so as you can judge of its quality and negotiate at kick off if needed.

If instead the management will be complex, using unusual life cycle, producing several iteration or prototypes, needing a complex organisation, requiring skills for support of new methods or training for new tools, if you want to monitor the technical budgets, then proceed with this process.

5.3 Software management

If you have selected this process, there are requirements that are always applicable (life cycle, phases, input, output, documents; technical specification phase and Preliminary Design Review [PDR], verification and validation process, interface requirement). Always select:	5.3.2.1	5.3.2.2	5.3.2.3	5.3.2.4	5.3.2.5	5.3.2.7	5.3.2.8	5.3.2.11	5.3.4.1
If the system interface are complex enough to deserve management procedures, select:	5.3.4.2								

If the system is real time and embedded, and you need to monitor the memory size, the execution time, the predictability and the performances, select:	5.3.5.1	5.3.5.2	5.8.3.11	5.8.3.12	5.8.3.13	5.8.3.14			
If the software development activities aim to deliver flight software (or ground real time software)	5.3.2.9	5.5.2.11							

Software Requirement & Architecture Engineering Process (5.4)

The **Software Requirement & Architecture Engineering** process is the first **Supplier** process. The purpose of the process is for the Supplier to write the Software Specification (this is called the *Technical Specification*) and the Architectural Design. This process is mandatory. Attempting to ask a Supplier to develop software without requirements is not professional.

According to your choice in the previous process, the Technical Specification may be merged with (or include) the Requirement Baseline.

5.4 Software requirements & architecture engineering process

Within this process, some activities are mandatory (functional and performance specification, data definition, external interface, unique identification of requirements, Preliminary Design Review [PDR]). Always select:	5.4.2.1-a)	5.4.2.1-e)	5.4.2.1-f)	5.4.2.3	5.4.3.14	5.3.3
If you need requirements about the system security (e.g. encryption), select:	5.4.2.1-c)					
If a human factors and ergonomics is important in the system, select:	5.4.2.1-d)					
If the software specification are complex enough to deserve a modelling, (e.g. static UML model, executable behavioural model) select:	5.4.2.4	5.4.2.5				
If you wish to have visibility on the architectural design of the software product, select:	5.4.3.1	5.4.3.2	5.4.3.3	5.4.3.8		
If those software requirements and architecture activities deliver a work product for flight or real time software, then select:	5.4.3.4	5.4.3.5	5.4.3.6	5.4.3.7		
If the software must be developed from reusable components, select:	5.4.3.11	5.4.3.12	5.3.2.15 (If 5.3 is selected: Not taken into			

			account in tool update)			
If the software is developed in order to be reused in other projects, select	5.4.3.9	5.4.3.10				

Software Design & Implementation Engineering Process (5.5)

The Software Design & Implementation Engineering process is the next Supplier process. The purpose of the process is to make the *detailed design*, the *coding*, the *unit testing*, the *integration testing*, and the *validation* by the Supplier against the Technical Specification. This process is mandatory as it addresses the development itself.

5.5 Software design & implementation engineering

Within this process, some activities are mandatory (coding). Always select:	5.5.3.1-a) 5.5.3.1-b)				
If you wish to have visibility on the detailed design of the software product, select:	5.5.2.1	5.5.2.2			
If those design and implementation activities deliver flight or real-time software, select:	5.5.2.3	5.5.2.4	5.5.2.5	5.5.2.6	5.5.2.7 5.5.2.11 5.3.2.9: already addresses in 5.3)
If you wish to have a formal unit test of the software modules, select:	5.5.2.9	5.5.3.1-c)	5.5.3.2		
If you wish to have a formal integration of the modules, select:	5.4.3.13	5.5.2.10	5.5.3.4	5.5.4.1	5.5.4.2

If you wish to have visibility on the problem reporting during the validation against the TS and the CDR, select <i>(May be to transfer in 5.8):</i>	5.8.3.10a				
If you wish to review the software design & implementation activities [detailed design, coding, unit test planning, execution and results, integration test planning, execution and results], then ask for a Critical Design Review [CDR] and select:	5.5.5.2	5.3.3	5.3.2.10 (if 5.3 is selected)		
If you need a Software User Manual	5.5.2.8	5.5.3.3	5.5.4.3	5.6.3.3	

Software Validation Process (5.6)

The software validation process is intended to confirm, at first, that the requirements baseline functions and performances are correctly and completely implemented in the final product and also to demonstrate the compliance to the software requirement specifications.

Within this process, some activities are mandatory (validation wrt RB). Always select:	5.6.4.1	5.6.4.2	
If the validation effort needs to be organized (planned, structured, sized), select:	5.6.2.1	5.6.2.2	5.6.2.4
If a validation against the Technical Specification [TS] (the specification of the Supplier, not your system specification) is needed, select:	5.6.3.1	5.6.3.2	
If you wish to review the validation wrt TS's tests results, ask for a Critical Design Review [CDR] and select:	5.6.3.5	5.3.3	5.3.2.10 (If 5.3 selected)

If the validation tests wrt TS are complex or large enough to be verified before the test campaign, or if the validation environment needs to be verified before the test campaign, then ask for a Test Readiness Review [TRR] and select:	5.6.3.4	5.3.3	
If the validation tests wrt RB are complex or large enough to be verified before the test campaign... , ask for a TRR and select:	5.6.4.4	5.3.3	
If you wish a dedicated Supplier review, in his own environment possibly simulated (Qualification Review), before your acceptance on the real environment, select: [Otherwise, QR and AR are merged.]	5.6.4.5	5.3.3	5.3.2.12 (If 5.3 selected)
If you think that some of the requirements cannot be tested e.g. because the test environment used for the software validation does not allow it or is not representative of the future real system environment	5.8.3.9		

Software Delivery and Acceptance Process (5.7)

The **Software Delivery and Acceptance** process is shared by the **Supplier** and the **Customer** (for the Acceptance). The purpose of the process is the *formal delivery and installation of the software product at the Customer premises*. This process is mandatory as it addresses your acceptance of the product. The Customer tasks of this process may also be delegated to the Supplier, but under the responsibility of the customer.

5.7 Software delivery and acceptance

Within this process, some activities are mandatory (delivery, installation report, acceptance test planning and execution, code regeneration, Acceptance Review [AR],). Always select:	5.7.2.1	5.7.3.1	5.7.3.2	5.7.3.3	5.7.3.4a)	5.7.3.4c)	5.7.3.5	5.7.3.6	5.3.2.13 5.3.3
If you believe the users need a specific training on the software product, to be developed by the Supplier, select:	5.7.2.2								
If the installation is so complex that it needs a plan and a report, select:	5.7.2.3	5.7.2.4							

Software verification process (5.8)

The **software verification process** generally consist **in a set of activities** that allow the review of the processes work output in order to check that they are complete, consistent, and feasible. If you have decided to dedicate a specific effort to the verification of the work outputs, in terms of independent reading, or inspection, because for example your system and software is critical, then you can select the relevant requirements.

<u>Requirements across processes: Verification activities</u>					
Always select	5.8.3.7	5.8.3.10			
Requirement Baseline verification, select:	5.2.4.4				
Software requirement verification, select:	5.8.3.1-b				
Software architecture and interface design verification, select:	5.8.3.2-b				
Software design, select:	5.8.3.3-b				
Software code, select:	5.8.3.4-b				
Software integration, select:	5.8.3.5	5.8.3.6			
Software validation against the Technical Specification, select:	5.8.3.7	5.8.3.8			
CDR verification	5.8.3.6				
Software validation against the Requirement Baseline, select:	5.8.3.6		5.8.3.8		
If this verification effort needs to be organized (planned, structured, sized), select:	5.8.2				

If the system is real time and embedded, and you need to monitor the memory size, the execution time, the predictability and the performances, select:	5.8.3.11	5.8.3.12	5.8.3.13	5.8.3.14	
--	----------	----------	----------	----------	--

The **traceability** is a particular verification activity, allowing verifying that the flow of knowledge on the system is preserved all along the development. The traceability from the requirement baseline towards the validation (against the RB) is mandatory and has been already selected. If you consider that other traces are needed, then select the appropriate requirement.

Traceability is useful at least for two purposes:

- To understand the impact of a modification during the maintenance, allowing to estimate cost, to define regression tests and to decide or not the change,
- To check the coverage of the a development phase compared to the previous, to make sure that nothing has been forgotten in the development, and to improve the test coverage at each step to improve the product validation.

Requirements across processes: Traceability activities

If the traceability from system requirements to software requirements is important, select:	5.8.3.1-a
If the traceability from software requirements to software architectural design is important, select:	5.8.3.2-a
If the traceability from software architectural design to software detailed design is important, select:	5.8.3.3-a
If the traceability from software detailed design to software code is important, select:	5.8.3.4-a

The **software product assurance** activities are fully defined in ECSS-Q80B, which is called in the following requirements:

Requirements across processes: Product assurance

The software product quality requirements [ECSS-Q80 7.2] select:	5.4.2.1.b)
Acceptance review, result of reviews and audit (ECSS-Q20) and validation testing (ECSS-Q80B 6.3.4), select:	5.7.3.4b

Software Operations Process (5.9)

The **Operator performs the Software operations process**. The Operator is the liaison between the Supplier and the User, in charge of making the system available to the User, performing the helpdesk, the database administration, the needed unblocking and reboot, etc. The Operator reports also the User problems to the Maintainer. The selection of this process depends on the coverage of your contract. Development contracts usually don't include operations. Simple software (e.g. using Install or Set-up on PC) usually doesn't really need operation.

5.9 Software operations

If the process is selected, some activities are mandatory (operation and user support). Always select:	5.9.4	5.9.5	
If the operations are complex enough to deserve a plan and a deployment, select:	5.9.2	5.9.3	5.2.6.1 [if 5.2 selected]

Software Maintenance Process (5.10)

The **Software Maintenance** process, performed by the **Maintainer**, consists in correcting the errors (corrective maintenance) or implementing changes (evolutive maintenance). This process is *always applicable* during the guarantee period (usually 6 months) indicated in the contractual conditions (clause 29). After the guarantee, development contracts don't always include long-term maintenance. The selection of this process depends on the coverage of your contract.

When the process is selected, three levels of formalism in the maintenance can be requested, depending on the budget, the level of risk and service acceptable for the user, the level of control needed by the customer, the type of application and its criticality.

- 1) A light maintenance with an informal analysis of the problem and a change implementation.
- 2) A nominal maintenance including in addition the reproduction of the problem and the decision to change in a CCB.
- 3) A formal maintenance including in addition a maintenance plan and formal changes and configuration management.

5.10 Software Maintenance

The light maintenance is the minimum mandatory activities of this process. Always select at least the light maintenance:	5.10.3.2	5.10.3.3	5.10.3.5	5.10.4.2	5.10.4.3
If you have selected the nominal maintenance, select:	5.10.3	5.10.4			
If you have selected the formal maintenance, select:	5.10.2	5.10.3	5.10.4	5.10.5	
If the software needs to be modified in flight, select:	5.2.7.2	5.4.2.2	5.10.2.5		
If a software migration is foreseen, select	5.10.6				
If a software retirement is foreseen, select:	5.10.7				

5 Clause 29 of contracts

The contract officer will include in the draft contract a clause for the acceptance or rejection of the software. It used to be adapted to PSS-05 and needs to be reworded. Here is a new version consistent with E40B:

CLAUSE 29: ACCEPTANCE AND REJECTION

Clause 29 is implemented as follows:

1. As regards documentation and reports, should the Agency's Technical Officer not accept the deliverables from the Contractor, he shall so inform the Contractor with the relevant justification. If no decision has been notified to the Contractor within one month of receipt by the Agency of the deliverables, the deliverables shall be considered as having been accepted.
- 2.1 As regards hardware, acceptance shall be performed in accordance with the Acceptance Procedure specified in section \ of Appendix 1.
- 2.2 For software required to be developed in accordance with the document ECSS-E40-Part 1 B, its acceptance shall be performed in accordance with the ECSS-E40 requirement 5.7.3.6.

Notwithstanding Clause 29.4 of the General Conditions, the acceptance report shall be replaced by the "AR milestone report [DJF;AR]" (expected output of 5.7.3.6.a). Upon signature by the two parties of the "Customer's approval of accepted state" (expected output of 5.3.2.13), the software shall be considered as provisionally accepted.

Upon provisional acceptance a warranty shall commence for a duration of 6 months. Final acceptance shall be notified by the Agency's nominated Technical Officer and shall be considered as the date upon which the warranty or any extension to it has elapsed or the date upon which all corrections to be performed in accordance with the warranty have been performed to the Agency's satisfaction, whatever date occurs the last.

- 2.3 For software not required to be developed in accordance with the document ECSS-E40 , acceptance shall be performed in accordance with the Acceptance Procedure specified in section \ of Appendix 1.
3. Rejected deliverables must be rendered compliant with the Agency's requirements and represented for acceptance within a time scale fixed in writing by the Agency.

6 Comparison between ECSS-E40 and PSS-05

If you need to convince yourself or your bidders that ECSS-E40 is not a revolution compared to PSS-05, then read the following.

Comparison ECSS-E40/PSS-05

(extracted from BSSC(98)2)

ECSS-E-40 describes the requirements for engineering space software and applications, i.e. **what** must be done. These requirements are based upon the requirements of ISO/IEC 12207 Software Life Cycle Processes [3, 4] and space system needs. In contrast, ESA PSS-05-0 describes **how** to engineer the software in terms of mandatory, recommended and guideline practices.

This report compares the:

- The ECSS-E-40 requirements with the ESA PSS-05 mandatory practices
- The ECSS-E-40 documentation requirements with the ESA PSS-05 documents and forms.

This report is expected to be of benefit to organisations that are currently applying ESA PSS-05 but are intending to apply ECSS E-40.

The conclusions of this report are that:

- No radical changes to existing practices are needed by ESA PSS-05-0 users when applying ECSS E-40
- The ECSS E-40 requirements not covered by ESA PSS-05-0 practices can be covered by adopting additional practices
- The ESA PSS-05-0 mandatory practices implement approximately 70% the requirements of ECSS-E-40.

The areas of requirements of ECSS E-40 that are not covered by ESA PSS-05 are:

- System architecture
- Software operations
- Software migration
- Software criticality analysis
- Reuse
- Man-machine interface.

Organisations that implement the requirements of ECSS E-40 by implementing ESA PSS-05 practices should develop practices to cover the areas identified above.

Users of ESA PSS-05 should note that ECSS E-40 does not:

- Define a software life cycle
- Contain detailed practices for specifying user requirements, software requirements and software designs
- Have a concept of provisional acceptance
- Require a Project History Document (although some of the contents of such a document are retained)

- Contain project management and configuration management requirements (these are in the ECSS M series of standards [5] to [12])
- Contain quality assurance requirements (these are in the ECSS Q-80 standard [14]).

(end of extraction)

As an indication, the following mapping can help to understand the new terminology:

ECSS-E40	PSS-05
Requirement Baseline	URD, IRD
Technical Specification	SRD, ADD, ICD
Detailed design	DDD
Verification and validation plan	SVVP/AD/DD/UT/IT/ST/AT
Development plan	SPMP/AD/DD/UT/IT/ST/AT
User manual	SUM, TRD