Chapter 8
Managing Software Process Evolution for Spacecraft from a Customer's Perspective

Christian R. Prause, Markus Bibus, Carsten Dietrich and Wolfgang Jobi

Abstract The Space Administration of the German Aerospace Center designs and implements the German space program. While project management rests with the agency, suppliers are contracted for building devices and their software. As opposed to many other domains, a spacecraft is a unique device with uncommon and custom-built peripherals. Its software is specifically developed for a single mission only and often controls critical functionality. A small coding error can mean the loss of the spacecraft and mission failure. For this reason, customer and supplier closely collaborate on the field of software quality. We report from a customer's perspective on how we manage software quality and ensure that suppliers evolve their processes: We contribute to standards, tailor quality, and process requirements to establish them in projects, and engage in cross-company product quality collaboration.

8.1 Introduction

The DLR is the national aeronautics and space research center of the Federal Republic of Germany. In addition to its own research, the DLR's Space Administration branch has been given responsibility for the planning and implementation of the national space program. It acts as customer and project manager during the making of hardware and software that it needs for executing its missions. The actual work of making is outsourced to external contractors.

The space sector is peculiar with respect to the fact that many spacecraft are one of the kind devices with uncommon and custom-built hardware and software. Scientific missions have no insurance; a second unit is never built. If the mission goal is not

C.R. Prause (✉) · M. Bibus · C. Dietrich · W. Jobi
Deutsches Zentrum für Luft- und Raumfahrt, DLR - Space Administration,
Königswinterer Straße, 522-524, 53227 Bonn, Germany
e-mail: christian.prause@dlr.de

M. Bibus
e-mail: markus.bibus@dlr.de

C. Dietrich
e-mail: carsten.dietrich@dlr.de

M. Kuhrmann et al. (eds.), *Managing Software Process Evolution*,
DOI 10.1007/978-3-319-31545-4_8

reached, for whatever reason, there is *no second chance*. Preparing a single mission and subsequent production of the spacecraft can take decades. Additionally, dependability requirements are very high because servicing hardware in flight is impractical to nearly impossible. Free-flying devices have to stay intact for decades under harsh environmental conditions. Software can potentially be updated in flight but whoever worked on 15-year-old software knows the troubles of maintaining aging software in a fast-paced technology field. Moreover, software often controls critical functionality. A single software failure can mean the loss of a spacecraft and its mission, for example, Ariane Flight 501 [11] or Mars Climate Orbiter [34]. Additionally, due to limited contact times with ground stations, uploading new software versions can take days. Therefore, higher efforts in avoiding software problems are justified [26, 28].

Project cost and time are nonetheless key topics: In the early 1990s, the NASA (United States' National Aeronautics and Space Administration) started its "Faster, Better, Cheaper" initiative; capping maximum project cost, reducing bureaucracy, and therefore enabling more parallel projects. The program put a high cost pressure on projects. Software became more important as a sponge for complexity, as "band-aid" for hardware design compromises [12, 28] and as a possibility to save on hardware and missions costs [36]. However, when several such light-weight missions failed, it became clear that it was necessary to reconcile speed with quality control [28]. In Germany, the Space Administration reacted to similar experiences [1] by significantly increasing its dedication to and efforts in hardware and software quality assurance activities [33]. Since that time, the product assurance department of the Space Administration is responsible for quality management in all major national space projects.

The agency's view on product assurance is dominated by the need for high quality and dependable products that result from novel and extreme technical challenges (Fig. 8.1), and the ever-new organizational contexts with the Space Administration's role as a customer without own making responsibilities but with a wide range of suppliers: Quality has to be—right from the start—built into the products that are provided by suppliers with highly diverse quality capabilities. Therefore, the suppliers' processes and their evolution are in the center of attention. Major challenges are, for example:

- Harmonize development processes at international level and across organizations
- Standardize tailoring to achieve consistent results and reduce subjective effects
- Check that software and software processes conform to applicable requirements
- Deal with suppliers' resistance to adapt their development processes
- Improve product assurance by exploring and introducing new methods and tools
- Evolve software processes to ultimately assure the quality of procured products.

This chapter describes the work of the Space Administration's software product assurance: Sect. 8.2 provides the context and background of this chapter. Section 8.3 introduces the ECSS system of standards, which is a joint effort of European space agencies and industries to harmonize their work. Section 8.4 explains how these standards are turned into a national catalog of quality and process requirements. Section 8.5 details the process of generating project-specific requirements from this

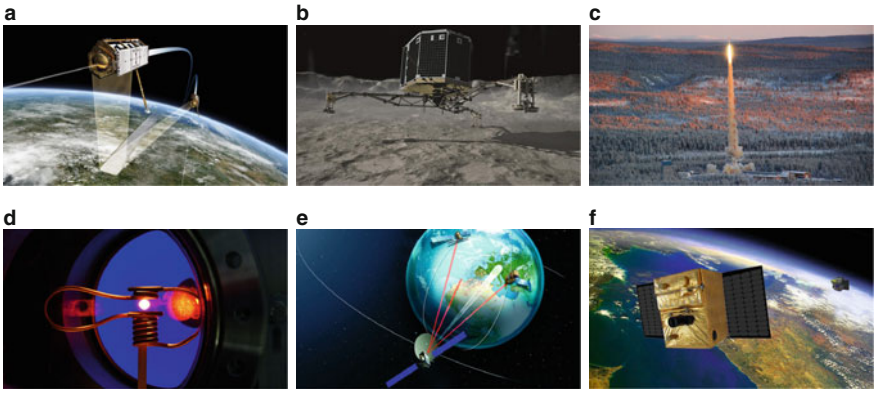

Fig. 8.1 **a** Radar twin satellites TerraSAR-X and TanDEM-X. *Source* DLR, CC-BY 3.0. **b** Philae touching down on Churyumov–Gerasimenko. *Source* DLR, CC-BY 3.0. **c** Rocket launch in the TEXUS Zero-G program in Kiruna. *Source* DLR, CC-BY 3.0. **d** Melting of materials without a container on board the ISS. *Source* DLR, CC-BY 3.0. **e** Laser communication terminal for inter-satellite and ground links. *Source* DLR. **f** TET-1 from FIREBIRD mission for detecting forest fires. *Source* DLR, CC-BY 3.0

national catalog through a standardized tailoring process. Section 8.6 outlines the responsibilities assumed by the DLR software product assurance during project execution including performance records, reviews, and technical visits. Section 8.7 gives an experience report on how a single process improvement was made possible against the initial resistance of suppliers. Finally, Sect. 8.8 concludes this chapter.

8.2 Background and Context

The DLR is Germany's space agency.[1] It consists mainly of distributed institutes that do research and development in the sectors of transportation, energy, flight, security, and spaceflight. DLR's Space Administration branch manages the German space program in the name of the federal government. It commissions devices (e.g., spacecraft) for its missions from a diverse range of suppliers including industrial and academic partners. It invites tenders, awards contracts for projects, supervises them afterward, and promotes innovative ideas in research and industry. As opposed to consumer products, devices are usually custom-built, expensive, and one-of-a-kind devices with high technical risk. Continuous customer, i.e., DLR, involvement in the process of making is therefore necessary. The role of the DLR Space Administration's

[1] *Note* ESA—the European Space Agency—is an international organization with currently 22 member states including Germany. DLR and ESA collaborate closely, and ESA committees include DLR representatives. Yet, they are separate organizations, both doing their own missions, having their own research divisions and mission operations, and procuring externally built devices.

Product Assurance department is to accompany technical processes in order to ensure product quality and successful completion of the project.

Product assurance is one of three primary project functions (the others being *project management* and *engineering*). It is a management discipline assuming the customers' viewpoint on product quality within the seller's organization. It supports project management in steering the product life cycle, and controlling production according to technical and programmatic requirements, while building on experience and lessons learned. Software product assurance disciplines include quality assurance with subordinate quality control, safety and dependability assurance, project planning, (independent) validation and verification, testing and evaluation, configuration management, and software measurement. Its functions are to observe, witness tests, analyze, and recommend, but not to develop or test, manage people, or set product requirements. Instead, it has organizational, budgetary, and product developmental independence meaning that it reports to highest management only, has its own budget, and does not expend labor to help building a product [6, 10, 32].

Customer product assurance mirrors the sellers' own product assurance function, acting as reviewer of contractors and technology providers ranging from large companies to small enterprises, research institutes, and universities. With respect to this, the function of product assurance is comparable to NASA's *Software Assurance Technology Center* (SATC; [3]). It assesses organizations and how they perform development activities in order to obtain software products that are fit for use and built in accordance with applicable project requirements [22]. For making sure that improvement (or evolution) of processes happens, enforcement is often necessary. This chapter makes a cross section through the several pillars of enforcement like relevant standards, contractual agreements, and active supervision (e.g., milestone reviews). It describes the managing of supplier software process evolution from a customer's perspective and through customer initiative.

Many organizations are usually cooperating in the production of a space device. They are bound by legal contracts in the roles of customer and suppliers which in turn act as customers to their lower tier suppliers. While the ECSS standards have no legal standing by themselves, they are made applicable by invoking them in the business agreements [15]. They provide a collection of what we call *process requirements* here (another term would be *software standards*). These are not to be confused with product requirements that describe what the product should do.

8.3 The ECSS Standards

Space technology is an extremely complex working field. In Europe, development and manufacturing of space systems is influenced by the cooperation of space agencies and industry since the beginning. One challenge of this work is the coordination of the use of compatible materials and implementation of compatible interfaces to reach quality and reliability as needed.

ESA developed *PSS* (*Procedures, Standards and Specifications*) standards to be applied in their projects for this purpose. Their use in projects of national European space agencies and industry had to be negotiated individually because national agencies developed standards individually and applied them to their projects. Rising demands made this approach more and more ineffective. Back in 1988, it was realized that there was the need of counteracting this trend [24].

In 1993 the European Cooperation for Space Standardization (ECSS) was founded to harmonize the requirements of existing standards for space projects and to develop and maintain a single, coherent set of standards for hardware, software, information management, and activities to be used in European space projects. The purpose of these standards is to continually improve the quality, reliability, functional integrity, and compatibility of all project elements.

The ECSS standards documents contain sets of requirements. Each requirement is verifiable, has a unique identification to allow full traceability and verification of compliance, and is supported by a minimal description necessary to understand its context. The documents themselves follow a systematic naming approach [15]:

```
ECSS-[branch]-[type]-[major][-minor][version]
```

where

Branch	One of the following values: P or S (ECSS system), M (management), E (engineering), U (sustainability), or Q (product assurance)
Type	The type is either ST (standard) or HB (handbook), which provides non-mandatory background information and reading help for a corresponding standard
Major	A two-digit number to identify the domain of the standard within its branch, e.g., *software engineering* (E-40) or *risk management* (M-80)
Minor	An optional two-digit number specifying a specialized substandard of a main standard, e.g., *ASIC and FPGA development* of the *Electrical, electronic and electromechanical components* domain (Q-60–02)
Version	A single letter from 'A' onwards for counting the major releases of the standards system (issue C at the moment)

The ECSS standards first and foremost focus on what has to be accomplished rather than on how to organize and perform the work. Interpretive help and details can, instead, often be found in the corresponding handbooks. This approach allows different producers and customers to apply established processes where effective as long as they remain within the fundamental constraints, and to improve and evolve processes gradually [15].

8.3.1 ECSS Policy, Members, and Organization

The ECSS policy is to develop and maintain an integrated and coherent set of management, engineering, product assurance, and space sustainability standards. The

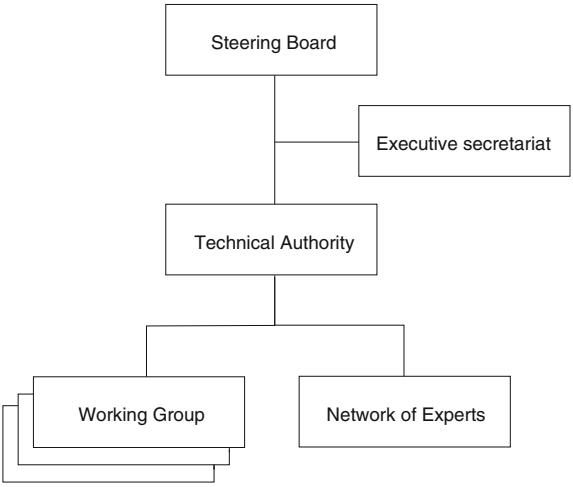

Fig. 8.2 The ECSS organizational structure [21]

objectives of ECSS are to increase quality, reduce risks, improve competitiveness, enhance safety and reliability, improve collaboration, and to develop and disseminate fresh knowledge. Principles supporting these objectives are, for instance, to seek harmonization with international standards by contributing to and ingesting from, e.g., ISO, CEN, and to continuously improve on the basis of user feedback. The standards are made freely available[2] to promote their wider usage [38].

The members of the ECSS are from European space sector (industry and space agencies) and associated organizations. They are differentiated between full members, associated members, and observers. Full members are those who actively participate in production, maintenance, and use of ECSS standards, like ESA, DLR, several national agencies (from France, Italy, the Netherlands, Norway, and UK), and Eurospace as representative of industry. Being an associated member (like Canada) indicates the desire to participate in production of ECSS standards and their limited application. Observers are those who desire to be formally informed about changes and be able to provide input in case of a need for an update or new standard. For instance, observers are the European Defence Agency or EUMETSAT.

The ECSS is organized in several bodies which also represent working levels (see Fig. 8.2). The top level is the ECSS *Steering Board* which defines ECSS objectives, policy, strategy, and endorses the yearly work plan. The *Technical Authority* implements the objectives, policy, and strategy defined by the steering board. It is also responsible for setup, approval, implementation, and monitoring of the work plan endorsed by the steering board. The elaboration of new and the maintenance of existing ECSS standards has to be performed by the *Working Groups* according to the work plan. Both Technical Authority and Working Groups are supported by the *Executive secretariat*, which enforces drafting rules, provides administrative support, and ensures promotion and interface with other standard development organizations. The lowest level is a *Network of Experts* representing document and discipline focal

[2] Visit http://www.ecss.nl/ for online access to the standards.

points, which give support to the Technical Authority and Executive secretariat in specific tasks [24].

8.3.2 Production and Maintenance of ECSS Standards

The development and update of ECSS standards are iterative processes (see also Fig. 8.3). A new standard is initiated by ECSS members sending a document called *New Work Item Proposal* (NWIP) to the Technical Authority, respectively, Executive secretariat. The proposal describes the envisioned content and scope of the standard, a justification for it, initial inputs, designated activities and milestones for Working Groups, necessary resources (e.g., composition of Working Group in terms of member organizations, manpower, and required meetings), and what the desired output is. It is then provided to ECSS members for public review to identify the need of extensions of the work item, and to recruit volunteer representatives from interested organizations. After the new work item is approved, the Technical Authority appoints the representatives to the Working Group, which then starts its work [21].

When the Working Group has prepared a draft version of the new standard, it undergoes a public review for comments. It is provided to all member organizations of the ECSS for this. All comments received are discussed by the Working Group and a first decision about their implementation is taken. The decisions are then communicated to the originators of the work item for agreement. Where no consensus can be reached, a final decision is taken by the Technical Authority. After this, the new ECSS standard is finalized [21].

If the need for an update of an existing ECSS Standards is identified, a formal *Change Request* is submitted to the ECSS secretariat. The next steps then are as

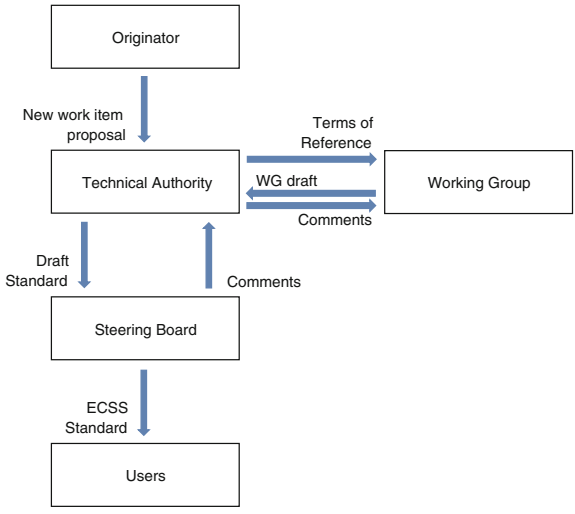

Fig. 8.3 Preparing new standard: information flow [14]

described above, starting with the preparation of a NWIP. If the Change Request highlighted problems without getting concrete on how to solve them, the Technical Authority calls for a special Working Group, a *Task Force*, to generate the NWIPs [21]. Through this process, European space projects have continuously returned feedback, corrections, and proposals to the ECSS standards and made the system extensive, efficient, and stable. The ECSS continues to be evolved and improved, also including the feedback processes themselves [38].

8.3.3 Software Standards in the ECSS System

Software pervades any space program and its product tree. For a concise overview of software development based on ECSS, see [29]. Apart from many general ECSS standards that are relevant to software development in space projects (e.g., configuration and information management [19]), there are several standards and handbooks specifically addressing software development.

The principles and requirements applicable to space software engineering are defined in ECSS-E-ST-40 (Space engineering—Software). Its first version appeared in 1999 as a specific adaptation of ISO/IEC 12207 to replace ESA's proprietary standard PSS-05-0 [29]. The current version takes the existing ISO 9000 family of documents into account and is in line with EN 50128 (railway applications) and DO-178 (airborne systems and equipment). It addresses development and use of software for manned and unmanned spacecraft, launchers, payloads, experiments and associated ground equipment and facilities, and services implemented by software. Covered aspects are software system engineering, requirements and architecture, design and implementation, validation, verification, delivery and acceptance, operation, maintenance and management. It also applies to nondeliverable software which affects the quality of products and services. The ECSS-E-HB-40A *Software Engineering Handbook* was created for daily use by suppliers. It provides advice, interpretations, elaborations, and best practices for the implementation of the requirements specified in ECSS-E-ST-40.

The software product assurance standard ECSS-Q-ST-80C [18] complements ECSS-E-ST-40C from the quality perspective. ECSS-Q-ST-80C interfaces with space engineering and management branches of the ECSS, and explains how they relate to product assurance processes. It is supplemented by five handbooks: ECSS-Q-HB-80-01A addresses the *Reuse of Existing Software*. The two volumes *Framework* and *Assessor Instrument* of the ECSS-Q-HB-80-02A *Software Process Assessment and Improvement* are known as *SPiCE for Space*, a software process maturity model derived from the SPiCE (Software Process Capability dEtermination) framework based on ISO/IEC 15504. Requirements regarding *Software Dependability and Safety* are further explained in ECSS-Q-HB-80-03A because dependability and safety are issues of paramount importance in the development and operations of space systems [20]. Finally, guidelines for *Software Metrication Programme Definition and Implementation* are provided through ECSS-Q-HB-80-04A.

Recently, the ECSS started to work on a handbook for agile development to live up to the growing interest in methodologies like Scrum and Extreme Programming. A respective standard is not yet planned due to a lack of consensus among partners regarding potential conflicts with established standards. Exactly this tension between agile and plan-driven development is further addressed in Chap. 2, where Diebold and Zehler also treat two approaches (evolutionary and revolutionary) for adding agility to plan-based processes.

8.4 Pre-Tailoring in the German National Space Program

Fig. 8.4a depicts different levels at which standardization can occur: international, regional, national, and company. International standards are usually rather specific in their scope. As opposed to this, the standardization process requires a very long time until consensus among diverse partners is found. On the other extreme are company-level standards. These in-house standards are the result of quickly made decisions among much more homogeneous parties, and they often cover a broader scope. Ideally, the different levels of standards are complementary, i.e., lower level standards only add details to a common, shared, higher level standard. In reality, however, the situation is not perfect. Standards at different levels overlap, and worse, sometimes contradict one another [24].

The reasons can be found, for example, in history (national space endeavors predate ESA), in corporate and national culture, in the different agencies' policies, or in different national interests. Likewise, German space industry differs from other

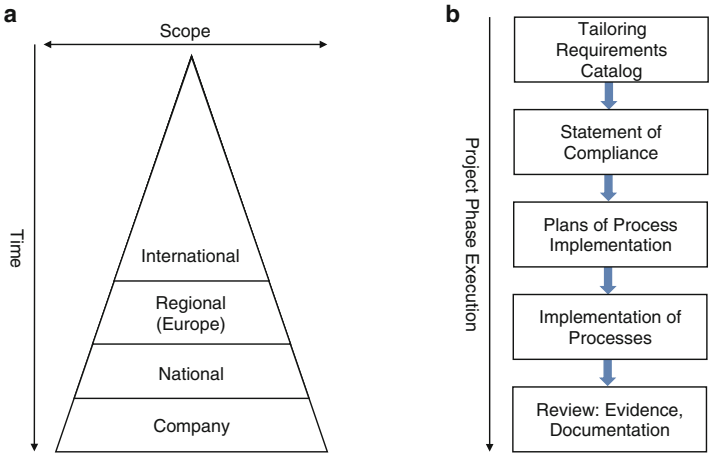

Fig. 8.4 **a** Time/scope relationship for different levels of standardization [24]. **b** Development process adaptation in project

European countries' space industry, and DLR differs from ESA. For example, conduction of advanced static analysis is a national requirement not found in ECSS.

So-called *pre-tailoring* is therefore regarded as necessary. It breaks the complex systems of standards down to national needs. These national needs are based on Germany's space strategy and cover also general project boundaries like environmental conditions, functionality, mission lifetime, and experiences from other projects. The standards considered for the pre-tailoring are national laws and standards, ECSS, ISO, military and NASA standards as well as requirements raised by the Russian space agency. The challenge of pre-tailoring is to identify requirements which are necessary for Germany's space projects but not contradicting rules and laws of other nations or organizations, which are partners. As an example, consider a project in which DLR delivers a payload for a satellite built by ESA: The industry contracted to manufacture the payload has not only to fulfill the requirements of DLR but also the ones from ESA. Due to the fact that Germany is no launch authority itself, the requirements of different launch authorities like Arianespace or NASA have to be considered, too.

8.4.1 Outline of the Pre-Tailoring Process

With *pre-tailoring* we denote a process that takes regional standards (ECSS) to turn them into a lower level but broader national catalog of requirements. The process is basically analogous to other tailoring processes as the 7-step process described in ECSS-S-ST-00C [15]. Of course, this tailoring is not yet aimed at a single project but at the virtual set of projects from the national space program. The steps are

1. Identification of possible types of projects and their characteristics.
2. Analysis of project characteristics with respect to cost, risk, technical drivers, critical issues, and specific constraints.
3. Selection of applicable ECSS standards as basis for pre-tailoring; standards referenced as applicable by selected standards become themselves selected.
4. Selection of applicable ECSS requirements from the selected standards by making a decision for each contained requirement.
5. Addition of new requirements specific to the national space program where ECSS is deemed lacking.
6. Harmonization of applicable requirements with respect to coherence and consistency of the overall set of requirements.
7. Documenting ECSS standards and requirements applicability.

On the one hand, by selecting requirements from several applicable standards at regional level, the scope of the resulting national standard broadens (as described for Fig. 8.4a). On the other hand, flexibility and time scale improve. For example, the ECSS is not yet harmonized with space standards from USA, Russia, or China [24]. On national level, however, consensus on integration can be reached easier.

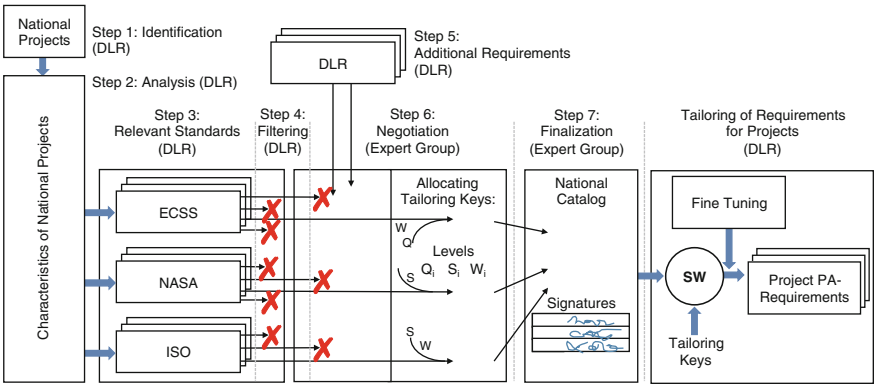

Fig. 8.5 Pre-tailoring, the national catalog, and computer-aided single-source tailoring

8.4.2 Pre-Tailoring Process Details

This section details the outline of the pre-tailoring process for the German national program (see also Fig. 8.5). Step 1 of pre-tailoring (in analogy to ECSS-S-ST-00C [15] tailoring) is to extract distinguishing characteristics from projects of the national space program. Relying on long experience, we selected several characteristics that distinguish projects in the national program. Such characteristics are, for example, the *Space Flight Type*, i.e., what kind of spacecraft, or *Utilization*, e.g., if the device is a free-flying satellite or used inside or outside of manned platforms like ISS or the space shuttle. In Step 2, these characteristics are complemented with refinement analyses regarding cost, risk, technical drivers, etc. Characteristics plus refinements form the dimensions of a nominal scale vector space. Any project can later be described as vector p consisting of these basic characterizations:

$$p \in C = C_{\{Sft,Lt,U,O,Mr,Sr,Fa\}} \times R_{\{L,B,C,Tr,Rp\}} \tag{8.1}$$

Step 3 of the pre-tailoring is to identify the relevant standards for software quality. First and foremost, this is the ECSS software product assurance standard [18] and parts of its complementary software engineering standard [17]. However, also requirements from other standards are integrated, for instance, configuration management (from [19]) or nonconformance reporting (from [16]).

In Step 4, every requirement is checked for its applicability to the national space program. For example, several requirements regarding software maintenance (ECSS-Q-ST-80C [18]; Sect. 6.3.8) were excluded because operations are out of scope of the space administration's product assurance.

Step 5 allows to add additional requirements to the catalog that are not yet considered by ECSS. For example, NASA and Roskosmos standards are necessary for ISS missions as these organizations are the safety authorities there. This step also allows to include novel processes not yet reflected in other standards (Sect. 8.7).

After the requirements have been gathered, they must be checked for internal coherence and consistency. Furthermore, it is necessary for later tailoring (Sect. 8.5) to discriminate more and less demanding process requirements. For this purpose, one or more requirement level tags are assigned to each requirement. Requirements imposed on software-specific processes are classified with one of four levels $w \in W_? = \{W_1, W_2, W_3, W_4\}$. The more rigorous requirement levels always include the all requirements of more light-weight levels: $W_1 \supseteq W_2 \supseteq W_3 \supseteq W_4$. The lower the level's number, the more demanding or expensive the process requirement is. For example, requiring the conduction of *Independent Software Verification and Validation* (ISVV) by a third party is at level W_1. It results in very high cost for the supplier. As opposed to this, software configuration management (W_4) is considered basic engineering rigor that should always be done.

In addition to software-specific process requirements, the software process is also influenced by cross-domain requirements from generic quality management and safety. They are classified analogously according to four Q levels, e.g., having a nonconformance control system in place is at Q_4. For safety, three S levels are defined, e.g., requiring conduction of software safety analysis is at S_3.

This sixth step is very work intensive and requires deep knowledge of the various requirements in order to assess their benefits, costs, effects, and cross-relations. It is therefore addressed by a work group of experts (Sect. 8.4.3).

Finally, as Step 7, the results of the requirements work group are documented in the product assurance requirements catalog [27]. They are additionally stored in a database used for automated tailoring (see Sect. 8.5).

8.4.3 Details for Step 6: Pre-Tailoring Expert Group

Every few years, the pre-tailoring expert group is convened to update the requirements catalog. It officially consists of one representative of every major stakeholder in the national program, i.e., from space administration and national industry. These formal representatives commonly have a *Head of Quality*-role or equivalent in their organization, and are supported by their domain/software expert. The expert group's input is an initial set of process requirements (partly from the previous version of the catalog) and its output is the national catalog. It enables

- to cope with the large amount of work associated with updating the catalog,
- to gather the necessary amount of practice and experience in one place, and
- to attain far-reaching justification and prominence for the resulting catalog.

Work starts by assigning so-called *field captains* to thematic subsets of the requirements, e.g., software product assurance, engineering management, or configuration management. The field captains then write a short review for each of their requirements. They see the item's ID, source, title, descriptive text, and requirement level tags ($W_?$, $S_?$). Additionally, they take into account aspects like cross-relations (duplications, contradictions, ...) or practical impact. They can propose a new title or text,

write a review comment, and propose a resolution of *accept*, *reject* or *modify* (*accept* means that the requirement should be included in the catalog, *reject* the opposite, and *modify* that first it should be changed in the specified way).

Next, all experts provide a comment and cast one vote on the proposed resolution, which can also be *accept*, *reject*, or *modify*. If all experts accept, the resolution is accepted. If at least one expert rejects the resolution or requests modification, the field captain has to make a proposal how to proceed. The proposal consists of an explanatory comment and a revised requirement text. All other experts again vote on the requirement using *accept* or *reject* as answers. If the majority accepts the modified requirement, it is included in the catalog. Otherwise, the requirement remains in conflict state until the decision is finally made in a round table discussion.

The expert group meets physically once when it constitutes and once it finishes. For the weeks in-between, work is supported by a web-based tool specifically developed for this purpose. It lets contributors view a list of all requirements along with key facts like if there is a controversy about resolution or if it is included in the catalog. The software furthermore tracks open points, allows contributors to write their reviews, cast their votes, and view details of the entailing discussions. Finally, it documents decisions and maintains the database with the catalog of requirements. The catalog is additionally printed as book and signed off [27] by all representatives to confirm its symbolic value.

8.4.4 Lessons Learned

Pre-tailoring enables a gradual, smooth, and careful while steady transition from traditional processes toward the ECSS standards. Over the years, the percentage of ECSS requirements reflected in the national catalog has constantly risen, reaching 43 % in 2008, 57 % in 2010, and 63 % in 2012. However, it also shows that national needs still differ from European ones.

The catalog also forms an agreed baseline for the national space industry. As the major players get their votes in the expert group, where they can veto against unreasonable process requirements, it gets more difficult for them to argue against those requirements later in the projects. Interestingly, only very few requirements are actually rejected through the expert group.

Pre-tailoring provides a consensual, objective, mission-independent balancing of benefits and costs of requirements because contents of the catalog are discussed decoupled from actual missions. It ensures that the selection of requirements in the scope of projects is not based on the personal preferences of the person who tailors the process requirements for the project, but is instead based on a systematic, repeatable, and standardized process. Decisions are made against the specific background and needs of the German space program.

In contrast to the ECSS, where DLR is only one partner, influence at national level as the leading customer is unevenly greater. It is much better possible to position and later realize software process improvements through requirements. We will come back to this later in the experience report in Sect. 8.7. The small group of experts allows faster decision making when meeting every few years, and is more open to try out not-yet widespread technologies.

The ECSS system is yet to be harmonized with the standards of the traditional space-faring nations like USA or Russia. But because Germany does not have its own launch capacities or sites, it needs freedom to choose its partners; and it therefore needs to implement foreign standards at national level.

Maintaining the catalog is a costly endeavor that should not be underestimated. Working through several standards each with hundreds of requirements takes its time. And the work needs to be repeated every few years in order to keep up with changes in the still developing ECSS standards system. Besides the discipline and endurance that are necessary anyways, the expert group is an important ingredient to dealing with the efforts. The web-based content management reduces the amount of required co-location time. It enables several people with densely filled appointment calendars to still collaborate.

8.5 Tailoring the Requirements for a Project

The ECSS is a system of coherent standards that supports a wide range of diverse space projects. In its original form, it might therefore not yet suit the individual project very much. This can result in reduced project performance in terms of technical performance, life cycle cost-effectiveness, or timeliness of deliveries [15], and is therefore considered as a major project risk (cf. [31]). In order to reduce this risk, tailoring is "the act of adjusting the definition and/or particularizing the terms of a general description to derive a description applicable to an alternate (less general) environment" [25]. *Tailoring* means fitting requirements placed on the process to the specifics of individual projects [15].

The basis for tailoring is the national catalog of product assurance requirements, as mentioned in Sect. 8.4. Three functions

$$f_W : C \to W_? \tag{8.2}$$

$$f_Q : C \to Q_? \tag{8.3}$$

$$f_S : C \to S_? \tag{8.4}$$

process the project vector $p \in C$ in order to obtain the applicable requirement levels. The requirement levels then select or deselect the individual requirements, resulting in the set of requirements applicable to the software development process. This tailoring is, for the most part, automated through the software tool *QMExpert Tailoring*.

8.5.1 The QMExpert Tailoring Tool

Tailoring the software process requirements for a new project begins with collecting the basic characteristics and analyses for the vector $p \in C$, where

$$C = C_{Sft} \times C_{Lt} \times C_U \times C_O \times C_{Mr} \times C_{Sr} \times C_{Fa} \times R_L \times R_B \times R_C \times R_{Tr} \times R_{Rp} \tag{8.5}$$

The basic characteristic dimensions and their values are summarized in Table 8.1.

Figure 8.6 shows the first input screen. The characteristics and their possible values are further explained at the bottom of the screen to ease the selection of the correct value: For example, the *Lander Spacecraft* in the *Space Flight Type* dimension is "designed to reach the surface of a planet and survive long enough to telemeter data back to Earth. ESA's Rosetta spacecraft [...] comprises a large orbiter, [...] and a small lander. Each of these carries a large complement of scientific experiments designed to complete the most detailed study of a comet ever attempted" [27].

The next step is to refine the choice of basic characteristics entered on the first screen according to analyses regarding the characteristics shown in Table 8.2. Here, for example, the *Technology Risk* value *low* means that for "the realisation of the product proven technology that are state of the art are available and can be applied" [27]. As shown in Fig. 8.7, the selections made here directly lead to the applicable requirement levels for $W_?$, $Q_?$, and $S_?$.

Table 8.1 QMExpert tailoring tool dimensions and values

	Name	Values
C_{Sft}	Space flight type	Robotic Maintenance System, Orbiter Spacecraft, Flyby Spacecraft, Lander Spacecraft, Rover Spacecraft, Application Satellite, Manned Flight, Military Spacecraft, Scientific Observatory Spacecraft
C_{Lt}	Launcher type	Expendable, Manned Reusable, Unmanned Reusable, Unmanned nonreusable Automated Transfer
C_U	Utilization	Free-Flyer, ISS internal, ISS external, Manned Launch Vehicle
C_O	Objective	Spacecraft, Payload
C_{Mr}	Maintainability requirements	Generic, Advanced, Complete
C_{Sr}	Safety requirements	Generic, Advanced, Complete
C_{Fa}	Flight authority	ESA, NASA, Roscosmos

Table 8.2 QMExpert tailoring tool project characteristics

	Name	Values
R_L	Lifetime	>7 years, 2–7 years, < 2 years
R_B	Budget	>50 M EUR, 25–50 M EUR, 10–25 M EUR, < 10 M EUR
R_C	Complexity	High, Low
R_{Tr}	Technology risk	High, Low
R_{Rp}	Risk policy	High, Low

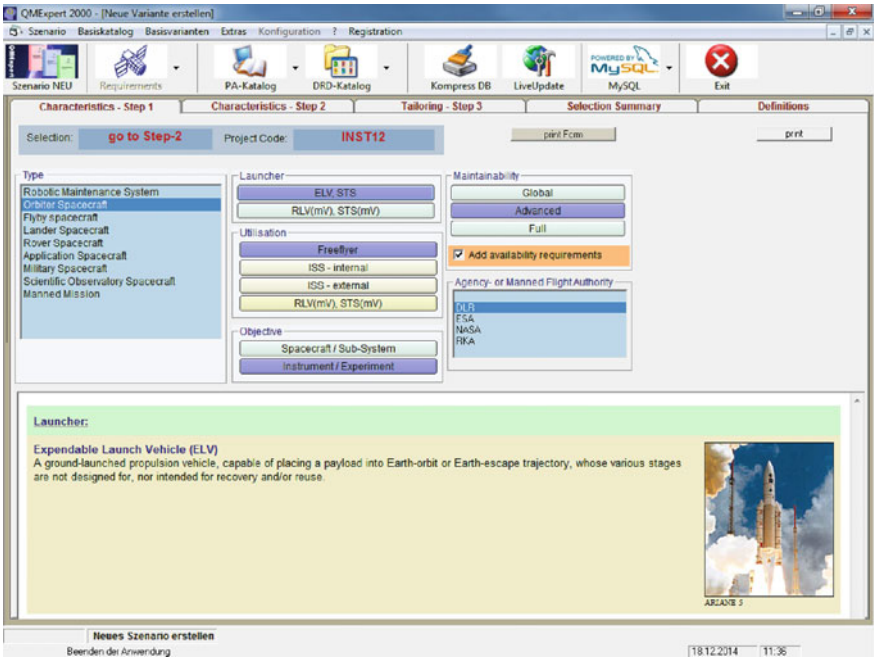

Fig. 8.6 Starting tailoring: screenshot of the project characteristics input screen

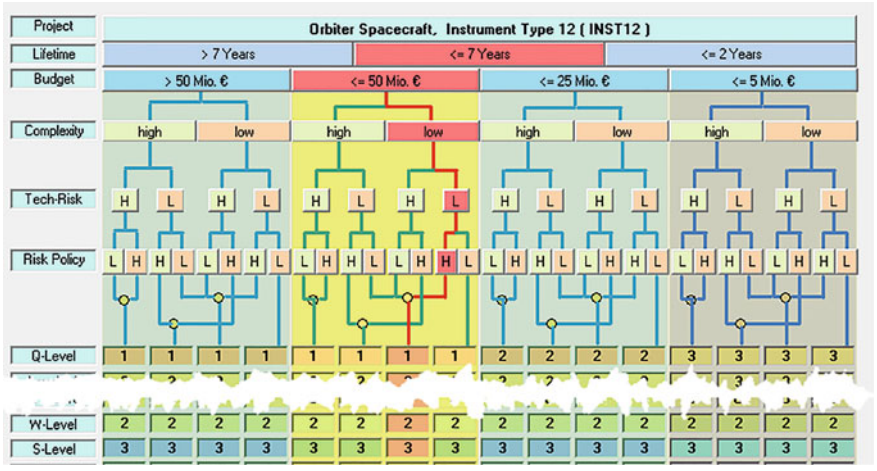

Fig. 8.7 Screenshot of adjusting tailoring parameters and resulting requirement levels $W_?$, $Q_?$, and $S_?$

After that, the tailoring tool picks the requirements for inclusion in the product assurance requirements document. As described before, all requirements in the catalog were tagged during pre-tailoring with one or more requirement level tags. For

example, the requirement that a hardware–software interaction analysis should be conducted is tagged with S_2 and W_3. It means that the requirement is included if the safety level is $s \leq 2$ or if the software level is $w \leq 3$, which is all applications that are safety critical or which have important software parts.

Next, the user is presented with a preview of the requirements, and an overview of which ones were selected and deselected. For fine-tuning, he can select additional requirements, or deselect requirements that he wants to be removed. The tailoring tool also ensures consistency by making sure that all requirements are included which are not themselves selected but which are referenced from other requirements. Finally, the tool exports into a Word document for further processing (e.g., including in contractual documents). Front matters, table of contents, abbreviation lists, chapters, and the like are generated automatically.

8.5.2 Lessons Learned

Tailoring is necessary for fitting coherent but generic standards to the specificities of a project. However, the national catalog contains hundreds of requirements applicable to software development. Manually tailoring them would be a huge effort, influenced subjectively by the tailoring product assurance manager's perceptions and emotions, and difficult to validate against corporate rules.

Our semi-automated tailoring process based on the QMExpert Tailoring tool is without frills but sophisticated. It is straight forward enough to be practicable. A single person can tailor a complete product assurance requirements document for a project in a short time. While manual intervention is still needed in several phases of the process, the tool significantly reduces the efforts for tailoring. Manual adjustments can be summarized in a report for validation by higher-ups.

Through the years, the tool has aged technically—it relies on dated libraries and technologies—but the process it supports and its contents have matured. A lot of tacit knowledge and experience went into the requirement level classifications, contributing to the quality of the tailoring results. Of course, much effort has been invested in the catalog data itself.

8.6 Cross-Company Product Quality Management

Unless a customer accepts any project result and quality, customer and supplier will seek *visibility* in order to mitigate the high risks of, for instance, untested technologies, large sums of money, or loss of life. Possible ways to achieve visibility are

- to negotiate contracts with intermediate products and partial payments, and
- to increase customer involvement in the development process [10].

8.6.1 Customer Product Assurance

Regarding intermediate products, space projects are executed in a series of *phases* cf. [29]. Each of the phases includes end milestones in the form of project reviews, the outcome of which determine payments and readiness of the project to move forward to the next phase. These reviews are the main interaction points between customer and supplier. Regarding customer involvement, the three primary project functions (project management, engineering, and product assurance) are present on the customer side as interfaces to their supplier counter-parts.

All three functions take their roles in ensuring the desired outcome of the development project: Project management is typically interested in getting the project out the door, thinking that engineers will take care of its quality. Engineers, however, are too concerned with getting the product to work that they will not see risks and potential weaknesses. The role of product assurance is that of a devil's advocate in a constructive and non-confrontational way. Product assurance benevolently probes the software product's contents. It has organizational and budgetary independence, and helps shaping but not building the product [10].

In the Space Administration, the product assurance department assumes the role of customer-side quality assurance for the procurement of space devices. It interacts with the supplier-side product quality functions, and primarily with quality improvement function in order to trigger improvements in the suppliers' development processes where necessary. Yet, the interaction between customer and supplier is not a one-way street: feedback, experiences, and knowledge generated from project execution is used to improve product quality management on the customer's side (Fig. 8.8). The toolbox of processes, methods, and tools for product assurance is continuously evolved. As a member of the ECSS standardization body, knowledge generated in the national program is forwarded further upstream and may eventually find its way into the ECSS standards system.

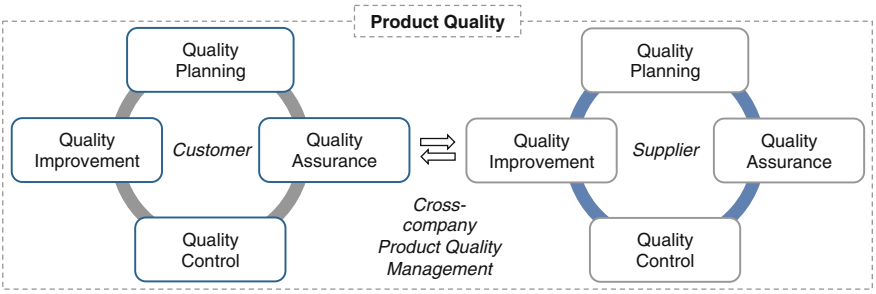

Fig. 8.8 Cross-company product quality assurance

8.6.2 The Implementation Process

Members of the customer product assurance are involved in project activities from the beginning. Software process requirements tailored from the national catalog (Sect. 8.5) are included in the contract as part of the work description. It is the foundation of product assurance work and defines objectives, policies, and rules for design, development, procurement, integration, and testing processes.

As part of the contractual negotiations, the supplier states his compliance to the prescribed development process requirements (see also Fig. 8.4b). The *statement of compliance* is a matrix indicating for each requirement the compliance status: either *fully compliant*, *partially compliant*, *non-compliant*, or *not applicable*. Unless a supplier declares full compliance with a requirement, the deviation and its reasons have to be explained in a commentary column of the matrix, and have to be accepted by the customer.

During the project, the supplier adapts its processes in order to comply with the requirements. For example, there is the general requirement of having established product assurance functions. While project management and engineering are commonly present on the suppliers' side, product assurance might be missing. Donaldson and Siegel [10] recommend to seriously question the maturity of such a supplier and its capability to ensure product delivery. In the national space program, however, the diverse small enterprises, universities, and research institutes often miss product assurance but still have to be involved for various reasons like promoting research and lack of alternatives. So one of the first process improvements is to establish product assurance.

As the requirements only prescribe what should be achieved but not how, the actual implementation is documented in respective plans, e.g., a Software Product Assurance Plan. The plans are reviewed at milestones for the customer to agree to their implementation. They serve to improve the visibility of the supplier's work, and are proofs of the implementation of requested processes. Besides milestone reviews, the customer's product assurance attends progress meetings, and looks out for deviations and defines the actions necessary to reach compliance. While most work is based on documents, the customer retains the right to visit a supplier's facilities any time and to perform inspections of work products.

In case a nonconformance is detected, product assurance participates in a *Nonconformance Review Board*, where further measures like root cause analysis, modification measures, and verifications are discussed and agreed upon. Typically, the supplier is capable of handling this by applying his quality management processes. If, however, the deviation's root cause is found to be in the supplier's processes, the deficiency is to be eliminated in the frame of process improvements.

8.6.3 Lessons Learned

The statement of compliance simplifies communication between customer and supplier by clearly summarizing the agreed-upon baseline of product assurance measures. It is part of the contract and often a major point of discussions and negotiations. Originally, suppliers only created plans in reaction to the requirements. This allowed them to more easily stretch requirement interpretations, and to better hide non-compliances. Through the statement of compliance, contradictions come up clearly and early in the project, reducing the risks of discovering them late.

Negotiating a statement of compliance that is accepted by both parties can be work intensive. Once agreed, however, it restates commitment of supplier project management to the development requirements. It is particularly valuable if a dispute arises during the project.

Attention should be paid to the understandability of comments in the statement of compliance because some projects last for many years. A change in personnel can mean that comments written too briefly may no longer be understood and cause new, unnecessary discussions. To reduce interpretive freedom and to avoid comments that negate a seeming compliance, we decided that comments (even explanatory ones) are not allowed for "fully compliant" responses.

8.7 Experience Report: Introducing Advanced Static Analysis

This section provides an experience report of how advanced static analysis was introduced in the German national space program. Static analysis is a widely used technology for detecting potential problems in software by analyzing human-readable or binary code without executing it. The ECSS prefers testing over static analysis for validation. But analysis is still recommended for verifying source code robustness and finding errors that are difficult to detect at runtime [17, 22]. The capabilities and complexity of static analysis techniques vary greatly from simple source code pattern analysis to formal methods including *abstract interpretation* [8].

In contrast to simpler analysis methods, tools based on abstract interpretation can prove the absence of several runtime errors (e.g., division by zero, arithmetic under- and overflows). Such a tool is also called *sound* [13]. One of the first commercially available tools capable of analyzing large code bases was Polyspace.[3] Compared to common simpler static analyzers, it is expensive with regard to financial cost and efforts. Annual license costs are tens of thousands of Euros plus one-time costs and initial trainings, and even on modern hardware analyses can run for hours.

[3] Available from: http://de.mathworks.com/products/polyspace/.

8.7.1 Polyspace Pilot Project

As a first step, a pilot project was set up. The purpose was to try out the capabilities of Polyspace, test if it will hold its promises, get a feeling for its handling, estimate if it is worth the cost, and generally build up expertise. The idea was to also try out, if Polyspace would fit into a toolbox for conducting software inspections as customer; metaphorically speaking, if it could be the software analogy to a magnifying glass a hardware customer uses when attending a *key inspection point* meeting.

For the pilot project, a Polyspace server was set up. One of the space projects that were just finishing volunteered to make available its satellite's flight software source code, which was about 22,000 lines of code. The software had passed all other validation and verification activities and was ready to be delivered. Next, it was imported into the Polyspace tool.

Polyspace decides for each line of code if the line is guaranteed to not contain the specified runtime errors (green), if the line will definitely cause a runtime error (red), dead code (gray), or if a decision could not be made (orange). The vendor forecast that in software of this maturity, Polyspace would still find about one runtime error per 1,000 lines of code. This forecast was met exactly. Consequently, several function-critical errors in the flight software could be fixed that might otherwise have caused serious troubles.

Yet, one drop of bitterness are the orange lines and computation time. The number of orange lines can be traded off against analysis computation time by adjusting the *precision* level. In our experience from other projects, rarely more than 20 % of lines are marked orange. This percentage and even lower values are also reported by other researchers [5]. Still, the undecided orange lines can cause non-negligible additional effort; in particular, as finding the root cause for a false positive (reported, but actually no error) located elsewhere in the code may require a thorough analysis.

Making rough estimates from the data provided in Emanuelsson and Nilsson [13], and Brat and Klemm [5], one can expect in 60,000 lines of fresh code: 40 error reports from tools like Coverity or Klocwork (both *unsound*), 9,000 orange lines from Polyspace, and 1,200,000 reports from FlexeLint (unsound) tool, which, however, can be tweaked down to 1,000 reports without thorough analysis. This means two things: First, unless one is willing to risk false negatives, i.e., missing out on certain errors, checking all suspect reports means a lot of work. So one better starts early. Second, a supplier may prefer to use an unsound tool in order to reduce the effort needed for checking suspects. The liability implications of knowing about potential errors (orange code) but not acting on them are, at best, unclear. But if the supplier did not know about the problems because he used an unsound tool, he can still plead research risks in case of an accident due to a software problem.

The pilot project showed that sound analysis is worth it because several critical errors were found in thoroughly tested code. However, it is not suitable for a quick inspection because major efforts are associated with importing the code into Polyspace, running the analyses, and checking orange code.

8.7.2 Toward Wider Adoption

The implication of the pilot project is that in order to reap the rewards of sound analysis, major efforts have to be invested in executing it. These efforts are beyond the capacity of customer software product assurance. Instead, industry should perform the analyses themselves. Only the reports were to be delivered for review. They serve as evidence that the analysis was executed, and allow to detect irregularities.

However, use of verification tools to demonstrate software quality is not explicitly specified in ECSS standards. Further, depending on the supplier, different tools are used. At the time we wanted to field Polyspace analysis processes, three projects were moving to their next phase. This meant that contracts (including software process requirements) were re-negotiated. Although the times were favorable, it turned out that introducing Polyspace was not simple. Separate and tiring negotiations for each project were necessary to place sound analysis. The space sector is conservative and dismissive toward changes to established processes. A common saying is "Only fly what has flown before!" The need for changing established processes and the costs associated with Polyspace (monetary license prices, and in terms of effort and legal risks) made it no surprise that industry would not easily agree. This holds true, in particular, if a process requirement is seemingly only imposed on a single project. But implementing a change from top-down through the ECSS standards seemed infeasible because consensus on multinational level would take many years and was further improbable to pass the respective committees without success stories.

In this situation, the convening of the pre-tailoring expert group offered the opportunity to implant the change on national scope. The invited top-level quality managers could be convinced of the net benefits of the sound static analysis, and without a concrete project in the background, the associated costs were too far away. In the end, the national catalog was extended correspondingly with a requirement regarding the proof of absence of several types of runtime errors. From there it gets tailored into requirements whenever a project moves to the next phase, and has nation-wide legitimation.

Meanwhile, sound static analysis is rather widely employed by suppliers. Even without being forced by a requirement, major suppliers have started procuring it for their other projects. However, every now and then, discussions still arise about sound static analysis during project execution. For example, if a supplier or one of its divisions are for their first time confronted with the need to provide the required report for a review. A supplier can then be pointed to the statement of compliance they signed (Sect. 8.6.2). If represented in the expert group, they can additionally be referred to the signing of the catalog by their head of quality (Sect. 8.4.3).

8.7.3 Lessons Learned

Conducting a pilot project first was important to learn that static analysis is a valuable addition to testing but also that it is not suitable as a tool for on-site inspections by the

customer. Instead, the verification itself has to be executed by the supplier according to contractual requirements. Evidence is provided in form of a report.

Strong rejection was a real problem initially. It was not practically feasible to overcome this rejection. This achievement was made possible only by establishing and exploiting the right management tools (pre-tailoring, expert group, and statement of compliance). Today, sound static analysis is broadly accepted by suppliers who worked with it. Only every now and then there is a new supplier or branch that has not worked with it. For these cases, the right management tools are in place.

The experience report provided here is only one example of a software process improvement. Further technologies are continuously researched and evaluated, and, if considered fitting, introduced into the national program.

8.8 Conclusion

In this chapter we presented software process evolution from the viewpoint of a customer. Our goal is to assure the quality of a product that is developed for a single purpose: to assume critical functions in a spaceflight mission. To reach our goal, we set the frame for development: We manage software process evolution through requirements from a strategic perspective, not how evolution is actually implemented by the providers organizationally. At that strategic level, we

- seek harmonization with ECSS and other standards,
- ensure implementation of process requirements at suppliers, and
- generate and disseminate knowledge to continuously advance processes.

However, it is difficult to harmonize and improve the processes in a sector with unequal histories and objectives, and diverse players. We revisited several levels at which the strategic frames for process evolution are defined, starting at the level of

- international and regional standards, moving on to the
- national catalog of the German space program, and further to the
- tailoring of process requirements at project-level, and the
- quality improvement efforts through cross-company quality management.

Taking the example of advanced static analysis, we described typical problems that can be encountered. It shows how the management tools at different implementation levels can be used to trigger process evolutions.

Many (if not all) organizations try to improve their processes by themselves. This, of course, is very important. However, they might have a different focus on what is important to optimize with priority. In a small market where products cannot be bought off-the-shelf but where products are unique specimen specifically developed for the customer, close collaboration between customer and supplier is necessary. Given visibility and trust, both sides profit from cross-company quality management.

8.9 Further Reading

Quality, software processes, and their improvement are an all-pervading topic in the knowledge areas of software engineering [4]. Our work[4] is distinguished from others through the fact that we describe how we address the evolution of software development processes toward higher quality from a customer's point of view. On the one hand, countless publications focus on organizations' work on improving their own software processes, e.g., [23, 26]. Doing so, indeed, is very important. On the other hand, much effort has been put into standards and maturity levels as a means of giving customers ways to assess the capability of suppliers. For instance, see ISO/IEC 15504 or CMMI [7, 30]. Furthermore, Rosenberg and Gallo [37] describe software product assurance at the NASA. However, not much has been published on the daily work of product assurance as a customer, and how software, tools, and methods improve this work.

In the space domain, tailoring of requirements to software development processes is omnipresent for aligning customer quality expectations with development effectiveness and efficiency [38]. Most ECSS standards already include a tailoring notice that explicitly encourages tailoring the standard. The ECSS-S-ST-00C further explains a formal tailoring approach based on the *ECSS Applicable Requirements Matrix*. It advocates putting all requirements with their identifiers in a table, and marking them as either applicable without change, applicable with modification, not applicable, and newly generated [15]. An adaptation of this approach is to include all requirements in their original form, and then record any changes or deletions after the original text, which, of course, can lead to very long documents. Currently, the ECSS is working on a more complex standard for the tailoring of ECSS standards.

ECSS-E-ST-40C and ECSS-Q-ST-80C are "self-tailoring." It means that both standards' annexes provide a table that lists for each requirement if it should be included in software with a certain criticality from A (most critical) to D (least critical). To determine the criticality category of each software item, a safety and dependability analysis and a hardware–software interaction analysis are conducted. The severity of the consequences of possible failures determine the criticality level.

A software tool for tailoring the ECSS-E-ST-40 Issue B is provided by the ESA. The wizard-style tool first takes the user through a questionnaire containing single-choice questions in several areas, e.g., project characteristics (novelty, complexity, expected lifetime, use of commercial off-the-shelf items, ...), stakeholders (who is the customer, supplier, maintainer, user, ...), risks (e.g., long-term use, tricky design), verification, and so on. It then outputs a table that proposes for each requirement in the standard whether it should be included or not. Döler et al. [9] presented a web-based tool capable of tailoring several standards including ECSS-E-ST-40B, ECSS-Q-ST-80B, DIN EN 50128 (railway applications), internal standards, and RTCA/DO-178B (airborne systems). Again characteristics like technical domain, software type, and operational complexity are queried using 17 nonredundant single-choice questions. The tailoring rules are based on comparisons with other standards and long-term

[4] An earlier version of this chapter was published as [35].

experience from working in space projects. However, both tools seem to be no longer actively maintained. Rumor has it that the tedious maintenance of the requirements and rule database might have been too costly.

Armbrust et al. [2] address product quality through scoping, i.e., what to include in a process and what not. Their approach is similar in that it characterizes space projects using criteria like mission type, complexity, or criticality that then result in adapted processes. Their view complements ours as it is technical and supplier-oriented: For example, cooperation with ESA triggered process evolution on their side.

Kalus and Kuhrmann [31] present a systematic literature review of criteria for software process tailoring. They identified 49 tailoring criteria, such as team size, project budget, project duration, the degree of technology knowledge, the availability of commercial off-the-shelf products, tool infrastructure, legal aspects, or the domain.

In Chap. 10, the authors describe an assembly-based method of process evolution. A focus of their work is the enactability and assurance of the enactment of activities imposed through regulatory needs or our requirements. With the same goal, Chap. 11 explains how to adapt case management techniques to deal with problems that stem from trying to achieve flexibility and compliance at the same time.

Chapter 13 addresses the co-evolution of development processes and model-driven engineering. They research the implied consequences for costs and success of process tailoring. This happens against the background of the importance of customization and optimization for staying efficient and dealing with arising new challenges.

References

1. Abbott, A.: Battery fault ends X-ray satellite mission. Nature **399**, 93ff (1999)
2. Armbrust, O., Katahira, M., Miyamoto, Y., Münch, J., Nakao, H., Ocampo, A.: Scoping software process models—initial concepts and experience from defining space standards. Making Globally Distributed Software Development a Success Story. Lecture Notes in Computer Science, pp. 160–172. Springer, Berlin (2008)
3. Basili, V.R., McGarry, F.E., Pajerski, R., Zelkowitz, M.V.: Lessons learned from 25 years of process improvement: the rise and fall of the nasa software engineering laboratory. In: Proceedings of the International Conference on Software Engineering, pp. 69–79. ACM, New York, NY (2002)
4. Bourque, P., Fairley, R.E. (eds.): SWEBOK V3.0—Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, Washington (2014)
5. Brat, G., Klemm, R.: Static analysis of the mars exploration rover flight software. In: Proceedings of the First International Space Mission Challenges for Information Technology, pp. 321–326 (2003)
6. Card, D.N.: Software product assurance: measurement and control. Inf. Softw. Technol. **30**(6), 322–330 (1988)
7. CMMI Product Team: CMMI for development, version 1.3 (2010)
8. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of the ACM Symposium on Principles of Programming Languages, pp. 238–252. ACM, New York (1977)

9. Döler, N., Herrmann, A., Tapper, U., Hempel, R.: Ecss application in dlr space projects—experiences and suggestions for enhancement. Presentation slides from the ECSS Developer Day at ESTEC (Noordwijk) (2005)
10. Donaldson, S.E., Siegel, S.G.: Successful Software Development, 2nd edn. Prentice-Hall, Upper Saddle River (2001)
11. Dowson, M.: The ariane 5 software failure. ACM SIGSOFT Softw. Eng. Notes **22**(2), 84 (1997)
12. Dvorak, D.L.: Nasa study on flight software complexity: Final report. NASA (2007)
13. Emanuelsson, P., Nielsson, U.: A comparative study of industrial static analysis tools. Electron. Notes Theor. Comput. Sci. **217**, 5–21 (2008)
14. ECSS Secretariat (publ.): ECSS—standardization objectives, policies and organization. ECSS Standard ECSS-P-00A, European Cooperation for Space Standardization (2000)
15. ECSS Secretariat (publ.): ECSS system—description, implementation and general requirements. ECSS Standard ECSS-S-ST-00C, European Cooperation for Space Standardization (2008)
16. ECSS Secretariat (publ.): Space product assurance—product assurance management. ECSS Standard ECSS-Q-ST-10C, European Cooperation for Space Standardization (2008)
17. ECSS Secretariat (publ.): Space engineering—software. ECSS Standard ECSS-E-ST-40C, European Cooperation for Space Standardization (2009)
18. ECSS Secretariat (publ.): Space product assurance—software product assurance. ECSS Standard ECSS-Q-ST-80C, European Cooperation for Space Standardization (2009)
19. ECSS Secretariat (publ.): Space project management—configuration and information management. ECSS Standard ECSS-M-ST-40C, European Cooperation for Space Standardization (2009)
20. ECSS Secretariat (publ.): Space product assurance—software dependability and safety. ECSS Standard ECSS-Q-HB-80-03A, European Cooperation for Space Standardization (2012)
21. ECSS Secretariat (publ.): ECSS—standardization objectives, policies and organization. ECSS Standard ECSS-P-00C, European Cooperation for Space Standardization (2013)
22. ECSS Secretariat (publ.): Space engineering—software engineering handbook. ECSS Standard ECSS-E-HB-40A, European Cooperation for Space Standardization (2013)
23. Falessi, D., Shaw, M., Mullen, K.: Achieving and maintaining CMMI maturity level 5 in a small organization. IEEE Softw. **31**(5), 80–86 (2014)
24. Gammal, Y.E., Kriedte, W.: ECSS—an initiative to develop a single set of european space standards. In: Proceedings of Product Assurance Symposium and Software Product Assurance Workshop, pp. 43–50. ESA (1996)
25. Ginsberg, M.P., Quinn, L.: Process tailoring and the software capability maturity model. Technical Report CMU/SEI-94-TR-024, Carnegie Mellon University, Software Engineering Institute (1995)
26. Holzmann, G.J.: Mars code. Commun. ACM **57**(2), 64–73 (2014)
27. Jobi, W.: Tailoring catalogue: product assurance & safety requirements for dlr space projects. Technical report, Deutsches Zentrum für Luft- und Raumfahrt (2012)
28. Johnson, C.W.: The natural history of bugs: Using formal methods to analyse software related failures in space missions. FM 2005: Formal Methods. Lecture Notes in Computer Science, pp. 9–25. Springer, Berlin (2005)
29. Jones, M., Gomez, E., Matineo, A., Mortensen, U.K.: Introducing ECSS software-engineering standards within ESA. ESA Bull. **111**, 132–139 (2002)
30. JTC 1 SC 7: Information technology—process assessment—part 1: Concepts and vocabulary. International Standard ISO/IEC 15504-1:2012, International Organization for Standardization (2012)
31. Kalus, G., Kuhrmann, M.: Criteria for software process tailoring: A systematic review. In: Proceedings of the International Conference on Software and System Process, pp. 171–180. ACM, New York (2013)
32. Ley, W.: Management von Raumfahrtprojekten. Handbuch der Raumfahrttechnik, 4th edn, pp. 715–764. Carl Hanser Verlag, Germany (2011)
33. Marsiske, H.A.: Wendepunkt Mars. http://www.heise.de/tp/artikel/6/6775/1.html (2000)

34. Oberg, J.: Why the mars probe went off course. IEEE Spec. **36**(12), 34–39 (1999)
35. Prause, C., Bibus, M., Dietrich, C., Jobi, W.: Tailoring process requirements for software product assurance. In: Proceedings of the International Conference on Software and System Process, pp. 67–71. ACM, New York (2015)
36. Rechtin, E.: Remarks on reducing space science mission costs. In: Proceedings of the Workshop: Reducing the Costs of Space Science Research Missions, p. 23ff. National Academy Press, Washington (1997)
37. Rosenberg, L.H., Albert M. Gallo, J.: Software quality assurance engineering at nasa. In: Proceedings of the IEEE Aerospace Conference, vol. 5, pp. 5:2569–5:2575. IEEE, Washington (2002)
38. Schiller, D., Heinemann, J.: ECSS—20 years of collaboration for european spaceflight. DLR Newsl. Countdown **24**, 32–35 (2014)