# Product Assurance of Software Reuse in the SPICE for Space Framework

João Gabriel Silva[Φ◇], Manuel Rodríguez[◇], Diamantino Costa[◇], Han van Loon[□], Patricia Rodríguez-Dapena[○], Knud Pedersen[×], Fernando Aldea-Montero[♦]

[◇] *Critical Software SA*
*Parque Industrial de Taveiro, Lote 48*
*3045-504 Coimbra, Portugal*
*{jgabriel, mrodriguez, dcosta}@criticalsoftware.com*

[□] *SYNSPACE AG*
*Hardstrasse 11*
*CH - 4052 Basel*
*hvl@synspace.com*

[○] *SoftWcare S.L.*
*C/ Serafín Avendaño, 18 Int.,*
*36201 Vigo, Spain*
*rodriguezdapena@softwcare.com*

[×] *Terma A/S, Space Division*
*Bregnerødvej 144, DK-3460 Birkerød, Denmark*
*knp@terma.com*

[♦] *ESA/ESTEC*
*Noordwijk, Netherlands*
*Fernando.Aldea.Montero@esa.int*

## Abstract

*This paper describes the main conclusions of the PA-PDS study[1]. This is a European Space Agency study about the reuse of Pre-Developed Software (PDS) in space projects, using S4S (SPICE for Space) as the framework. The main objective of this study is to define the product assurance requirements to support the acquisition, evaluation, integration and maintenance of PDS to be reused in a new development of a space system. The method proposed for reuse of PDS is described, particularly the main requirements to follow. The project has considered technical and organisational viewpoints. Both the perspective of occasional (or informal) reuse and systematic reuse are addressed.*

## 1. Introduction

Developing large industrial systems with very high reliability and availability requirements entails enormous costs. This is the reason why many organizations have begun to consider implementing such systems using reusable component repositories. In the framework of the space domain, the European Space Agency (ESA), like other government and system developers acquiring software-intensive space systems, are undergoing a shift in emphasis from custom-developed software[2] towards the use of *Pre-Developed Software* (PDS)[3], with the expectation that:

- It can significantly lower development costs and shorten development cycles.
- It can lead to software space systems that require less time to specify, design, develop, test, and maintain, yet satisfying high reliability and quality requirements.

To achieve this, it is important to define new requirements and processes within the European space standards addressing the development of software-intensive systems. That is the main motivation and purpose of the PA-PDS study, which has approached the above observations by focussing on the product assurance aspects required for making development 'with' reuse[4] a success.

The product assurance areas that the study has targeted are the following:

- Software lifecycle development phases
- Software quality models, metrics and evaluation methods
- Software product evaluation and certification
- Risk management
- Safety and dependability requirements and techniques

---

[Φ] João Gabriel Silva is a professor at the University of Coimbra, Portugal, acting in this project as a senior consultant to Critical Software.

[1] PA-PDS ("Product Assurance Support to Pre-developed Software Proposed for Reuse") is a study sponsored by the European Space Research and Technology Centre (ESTEC) of the European Space Agency (ESA) - ESTEC Contract 15808/01/NL/WK.

[2] *Custom-developed software*: Software that is specifically developed under the framework of a project.

[3] *PDS*: Any software component or asset that has been developed outside the context of the project where it is being considered for reuse, such as a commercial operating system or a document template.

[4] In order to make affordable the scope of the PA-PDS study, neither development 'for' reuse nor domain engineering were considered. Also, other issues not directly related to technical aspects were left out of scope, in particular legal and contractual concerns.

- Process assessment (as extensions to SPICE for Space - S4S[5])

A number of technical notes have been produced under the project, which will be available at ftp://ftp.estec.esa.nl/pub/tos-qq/qqs/PDS/. Hereafter, we provide a brief description of the scope of these technical notes:

- Characterization of PDS for reuse in space projects and classification of PDS (TN1 [3]).
- Survey and analysis of industry and research literature – SURPRISE, McClure, ESI-Reuse, SEI PLP, etc. –, and software standards – IEEE, NASA, ISO, ECSS, SPEC, etc. (TN2 [4][6]).
- Creation and extension of processes, base practices and work products in S4S that consider software reuse (TN3.1 [5]).
- Definition of product assurance requirements and guidelines for reuse (TN3.2 [6]).
- Technical specification of tools supporting reusable component repositories (TN3.3 [7]).
- Pilot projects to demonstrate the applicability of the results of the PA-PDS study, based on the reuse of SCOS 2000 [15] and OBOSS-II [18] in the Herschel-Plank satellite missions (TN4 [9] and TN5 [10]).
- Proposed changes to the European Co-operation for Space Standardisation (ECSS) standards, the Software Product Evaluation and Certification (SPEC) method [13] and other ESA related studies [8].
- Tutorial materials for managers and technical personnel both from industry and ESA [11][12].

The main contributions of the PA-PDS study are:

- Definition of Product Assurance (PA) requirements and PDS types to support the acquisition, evaluation, integration and maintenance of Pre-Developed Software to be reused in a new space system development. This contribution is basically covered by technical notes [3] and [6].
- Reuse process assessment additions to SPICE for Space (S4S)[7]. This contribution is basically covered by technical note [5].

The second contribution (the proposed extensions to S4S) was already described in a previous paper presented in the SPICE conference [1].

This paper focuses on the first contribution of the project, namely the definition of product assurance requirements. They naturally depend on the circumstances of reuse, because it is not the same to reuse e.g. a real-time kernel that runs on a satellite, or a mathematical routine that processes off-line data received from that satellite. We call these different reuse scenarios PDS types (they are described in section 2) and indicate for each individual requirement to which PDS type does it apply. The defined requirements also take into account organisational concerns when PDS is used, and complement the ISO12207 organisational process requirements. They also target different groups, including software engineers, product assurance personnel and project managers, within industry as well as within ESA.

The paper is structured as follows. Section 2 introduces the classification of the different PDS types. Section 3 presents the fundamental (or *top*) requirements that allow an organisation to move from no reuse to occasional reuse to systematic reuse. This transition is necessary so as to achieve a significant benefit from PDS reuse. Section 4 covers the main organisational issues that allow for such a transition. Finally, Section 5 concludes the paper and gives future directions of this work.

## 2. Pre-Developed Software types

An important activity of the PA-PDS study consisted of establishing a characterisation of the existing PDS, and defining a set of reuse types.

The characterisation of PDS was made according to various attributes, which were grouped into four global categories called *perspectives*. These are described hereafter:

1. *Space domain perspective*. The analysis of the domain brought forward four major types of systems where the PDS could be part of: spacecraft/on-board, ground segment, EGSE[8], and support software.
2. *Criticality perspective*. It corresponds to the criticality level of the functions to be implemented by the PDS. This is based on an overall hazard analysis of the overall system. The rational for the different criticality levels is given hereafter:
   - Criticality level A: Software potentially leading to 'catastrophic' consequences (loss of life). For example, the collision avoidance software in a man transfer vehicle.
   - Criticality level B: Software potentially leading to critical consequences (loss of mission, major damage to the system). For example, the attitude and orbit control software of a launcher vehicle.
   - Criticality level C: Software potentially leading to major consequences. For example, the entire on-board software of a communication satellite.

[5] SPICE for Space (S4S) is an extension of ISO 15504 to include requirements specific to the space industry [14].

[6] Since this technical note is just an intermediate result leading to [5][6], it will probably not be made available in our FTP site, but can be provided under request.

[7] These additions are only proposed extensions (their inclusion has not been decided upon yet).

[8] Electrical Ground Support Equipment

- Criticality level D: Software potentially leading to significant consequences. For example, the ground software for data reception and archiving of a satellite instrument.
- Criticality level E: Software potentially leading to negligible consequences. For example, most off-line software, like ground software to process scientific data.

3. *Level of reuse perspective*. It corresponds to the different levels at which reuse may be applied, namely: reuse of basic operations (e.g. mathematical functions), reuse of data types (e.g., stacks and lists), reuse of utilities (e.g. a Kalman filer), and reuse of entire subsystems (e.g., attitude and orbit control software of a spacecraft).
4. *Reuse perspective*. Three reuse perspectives are considered: (i) commercial aspect (e.g., in-house developed software, third party software), (ii) types of component (e.g., design documentation, source code, object code), and (iii) intended reuse (e.g., stand-alone component, integrated 'as is', embedded in hardware).

From these four perspectives and the corresponding attributes, multiple combinations are possible that lead to different PDS types. The PDS types are meant to tailor the proposed reuse requirements [6]. After several iterations, we arrived at the four PDS types presented in Table 1, which we consider to be the relevant ones for the tailoring of requirements.

**Table 1.** PDS types

| ***Perspectives*** | | | ***PDS types*** |
|---|---|---|---|
| *Criticality* | *Space domain* | *Source code available* | |
| A/B/C | On-board /Ground SW | Yes | **1** |
| | | No | **2** |
| | Support SW | - | **3** |
| D/E | All | - | **4** |

It is worth noting that the PDS types ordering presented in Table 1 does not imply any ranking.

According to Table 1, the reader will notice that from the last two perspectives considered in the beginning (i.e., level of reuse and reuse perspectives), only the COTS[9] attribute (from the reuse perspective) has a significant impact on the tailoring of the requirements. Therefore, the key attributes for the definition of the PDS types are the criticality level, the space domain, and the availability of source code. The availability of source code is a key criterion when deciding what activities to apply to PDS compared to custom developed software (test suites, safety and dependability analyses, etc. – see requirement #1 of Section 3). This attribute is more relevant than other attributes (e.g., public domain vs. commercially supported, configurable software vs. a piece of code to be reused 'as is'). Indeed, in order to validate most space systems, the source code is needed so as to check issues like 'dead code'[10], unreliable software constructs, or timing and performance issues, and perform various verification and validation activities like software inspections.

Hereafter, we provide a rationale for each PDS type of Table 1:

- PDS types 1 and 2. They consist respectively of on-board and ground software of criticality levels A, B or C, whose source code may be available or not. The failure of such software may lead to catastrophic or critical consequences. For on-board software, it will typically consist of software of small size, with limited reuse. For ground software, it includes software development tools, whose outputs are of criticality level A, B or C. These tools should be of the same criticality level as the software they are generating. Examples of PDS of type 1 are the Spacecraft Control and Operation System SCOS 2000 [15], as well as the RTEMS [16] and ORK [17] operating systems. An example of PDS of type 2 consists of the CTREE component of SCOS 2000.
- PDS type 3. It consists of support software of criticality level A, B or C. PDS belonging to this type may be a publicly available case tool for which object code (and maybe also source code) is available. Examples are compilers, linkers, and loaders used to generate onboard software of criticality level A, B or C (e.g., the GNAT/ORK compiler, which is used for on-board software).
- PDS type 4. It consists of all software of criticality level D or E. A large spectrum of PDS belongs to this group. Examples are the software validation facility (SVF) tools used to perform independent software validation. These tools are meant to find errors in onboard software (although they cannot introduce errors, they may fail to discover an error).

As already stated, the proposed reuse requirements [6] are tailored according to these PDS types. Figure 1 presents a straightforward example of this tailoring.

The next section presents a summary of the most important (or *top*) requirements.

[9] *COTS*: In the context of this study, COTS is defined as a software component whose source code is not available.

[10] *Dead code*: It is executable object code (or data) which, as a result of a design error, cannot be executed (code) or used (data) in a operational configuration of the target computer environment and is not traceable to a system or software requirement.

4.2.4.1.8

| Requirement | PDS type | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Assets **shall** be assessed from the safety point of view (when applicable). | | Y | Y | Y | |

**Figure 1.** Tailoring per PDS type of a requirement

## 3. Top requirements

This section presents the fundamental (or *top*) requirements that allow an organisation to move from occasional reuse to systematic reuse. This transition is necessary so as to achieve a significant benefit from PDS reuse.

A major conclusion drawn from the survey of existing practice within space projects, was that reuse is performed occasionally or informally, rather than systematically. Indeed, the reuse that has occurred to date in space contractors is quite often performed through the 'reuse' of personnel from other (earlier) projects, who informally select and reuse various items, such as templates, plans, documents or code. This same mechanism has also led to the reusing of other types of items like test suites, safety and dependability analyses, design documentation, etc.

This informal practice of performing reuse does not allow for the systematic reduction of time-to-market and development costs. This conclusion forms the basis for how the top requirements are presented in Figure 2: they are shown against an implementation scale from occasional to systematic reuse.

| | | |
|---|---|---|
| Occasional reuse | #1 | PDS shall be applied the same product assurance activities applied to custom developed software |
| | #2 | Black-box PDS shall be avoided for criticality levels A and B |
| | #3 | The deactivated[11] and dead code[10] of a reused PDS shall be controlled/removed |
| | #4 | Reused software shall consist of much more than just code |
| | #5 | In-service history shall be used to tailor requirements |
| Systematic reuse | #6 | Reuse shall be considered when system and software requirements are defined |
| | #7 | Reuse shall imply new processes at the organisational level |

**Figure 2.** Requirements reflecting the transition from occasional to systematic reuse in an organisation

The successive application of these requirements is meant to help make the transition from occasional reuse to systematic reuse. In the sequel, we give the rationale for each requirement.

As stated by requirement #1 of Figure 2, the driving idea behind a method for reusing software in space projects, is that one should apply to PDS exactly the same activities that are applied to custom developed software, be it verification and validation, risk assessment or quality/certification metrics measurement. This is a direct consequence for instance of clause 6.4.3.1 "Analysis of potential reusability" of the space standard ECSS-E40B [2], and results from the fact that people are not willing to compromise on the quality assurance methods applied, as the reuse scenario is never exactly the same as the one where the reused asset was originally developed, and even tiny differences can lead to big disasters, as the accident in the maiden flight of Ariadne 5 has clearly shown. It might seem at first analysis that this jeopardizes the potential benefit of reuse, but that is not necessarily so, as also the support documentation required to apply those quality assurance methods can largely be reused, as stated in requirement #4, described below.

Requirement #2 of Figure 2 states that black-box PDS[12] as operational SW should be avoided for criticality levels A and B (see Section 2 for the definition of the different criticality levels). If such PDS does have to be used at all, then an analysis of possible failures shall be carried out and a strategy shall be defined so as to detect failures of the black-box PDS and protect the system from these failures, e.g. through wrapping by code that intersects all communication between that asset and the outside world, and filters misbehaviours. The protection strategy shall also be subject to validation testing. Error logs shall exist and shall be evaluated. As far as practicable, only the simplest functions of the black-box PDS shall be used.

Special cases (requirement #3 of Figure 2), such as unreachable code or deactivated code[11] of a PDS shall be carefully controlled. Unreachable code should only be allowed to remain in the final application where it can be shown that the risks of leaving it in are less than the risks of modifying the code to remove it. Deactivated code shall be disabled for the environments where its use is not intended, or removed when used in critical software. An analysis should be performed to assess both the effect of such a removal and the need for re-verification. Removing dead code or unused variables allows the design errors that caused them to be recovered from in the final software product. It also precludes its inadvertent execution, which may result in a system hazard.

The most interesting software to be reused should consist of much more than just code (requirement #4 of

[11] *Deactivated code*: It is executable object code (or data) which by design is either (a) not intended to be executed (code) or used (data), or (b) only executed (code) or used (data) in certain configurations of the target computer environment.

[12] *Black-box PDS*: Assets for which the source code is not available to the reusers.

Figure 2). The reused PDS should consist of a package containing not only code, but also design documentation, test suites, safety/dependability analyses, quality metrics, etc. Otherwise, the cost savings may be significantly lower than initially expected, or not relevant enough to justify reuse. Note that the customer will be contrary to the idea of accepting intensive reuse of software in highly critical systems unless it is provided with enough safety and dependability evidence. An interesting conclusion was drawn from the pilot projects of the study [9][10], which is related to this requirement. The pilot projects demonstrated that reusing PDS not originally developed to be reused, is of little profit and can even be counter-productive, due to the need to perform a comprehensive extra set of activities (e.g., provision of missing safety evidence or functionalities by significantly modifying the PDS).

In-service history (requirement #5 of Figure 2) can be used to tailor project requirements, especially when it is not possible to satisfy them directly (e.g., due to the lack of source code or design documentation) or when significant costs may be saved. However, in-service history always requires negotiation between the developers and the customer/certification authority/system safety responsible. In particular, it should be determined whether the previous usage profile of the candidate PDS is relevant enough to the reuse scenario.

To benefit from reuse, there needs to be a systematic consideration of reuse aspects during the requirements specification phases of a project lifecycle (requirement #6 of Figure 2), which in European space projects is normally the responsibility of ESA. At least, flexibility for waivers should exist. The customer (e.g., ESA) should be ready to accept non-compliances to accommodate reuse aspects on issues like functional requirements (e.g., performance) and non-functional requirements (e.g., design/programming language, V&V tools). During this negotiation, the supplier should demonstrate that the acceptance of non-compliances is also profitable for the customer (e.g., because of a reduction of effort and development costs). The development organisation has also to be adapted for development with reuse, as it has different needs than custom development. The European Space Agency, being interested in reuse, should foster it from the beginning (e.g., internal project proposals), explicitly including reuse at the system requirements level and being flexible enough to accommodate a reuse offer (e.g., by considering how requirements can be waived/adjusted).

Systematic reuse needs the definition of new processes at organisational level (requirement #7 of Figure 2). The organisational processes considered are illustrated in Figure 3[13].

[13] These organisational processes have been adapted from ISO12207/AMD2002 [19] and IEEE 1517 [20].

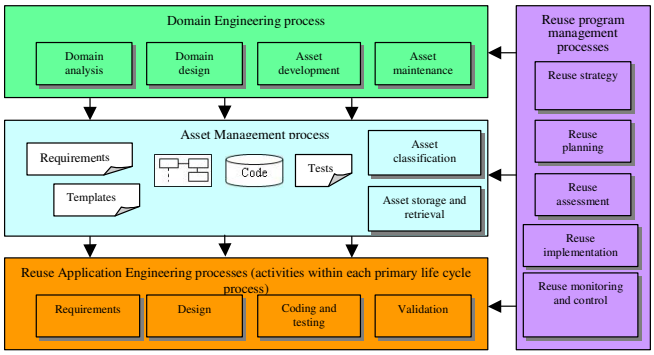


**Figure 3.** Reuse processes within an organisation

- *Reuse program management process* (for reuse at the organization level). This process is used to plan, establish, manage, control, and monitor an organization's reuse program (see Section 4.2 for more details).
- *Asset management process* (for assets libraries). Asset management is the process of applying administrative and technical procedures throughout the life of an asset (see Section 4.3 for more details).
- *Domain engineering process* (for assets definition). The purpose of this process is to develop and maintain models, architectures and assets for a particular domain. It includes the acquisition, development and maintenance of assets belonging to the domain. Note that this process was out of scope of the PA-PDS study (see footnote 4).

These processes form the basis of the proposed extensions to the S4S model for reuse, which is the second main contribution of the PA-PDS study [1].

Note that Figure 3 also contains other processes, namely the *reuse application engineering processes*. These correspond to generic engineering activities that should be performed at every project lifecycle phase, so as to consider the potential reuse of any kind of asset (e.g., requirements, documents, templates, code, etc.). These activities consist of searching for assets, selecting assets, assessing assets, integrating assets, and providing feedback.

Next section provides more details about the fundamental organisational issues.

## 4. Organisational issues

This section describes the organisational issues an organisation needs to consider to fully achieve systematic reuse of PDS. It comprises procurement organisation, reuse program and asset management system. The related roles and actors are also described.

## 4.1. Procurement organisation

For big projects with a complex project organisation, e.g. with multiple subcontractors, as is usually the case in space industry, the following issues shall be considered when planning for the reuse of PDS components:

- The need for a global or individual procurement planning at the programme level.
- The need for a project procurement plan for each software project.
- The opportunity to develop in common the qualification package when the same software product is acquired in several software projects.

The procurement planning should document the programme policy for software procurement. Such planning should be elaborated as early as possible and should be submitted for review at system level. The rationale for this approach is to be able to share information about procurement of software products in the whole programme in a controlled way. It should also contain up-to-date information related to the certification status of the considered products (e.g., safety analyses).

The project procurement planning consists of a specific plan for each type of product to be acquired at the project level. Such a plan should specify the process to be applied for the acquisition of the needed software product. When tool qualification or software safety assessment is needed, the analysis of its feasibility should be planned as early as possible in the procurement process. The actors of this possibly centralised PDS acquisition or procurement process within a project are the prime contractor (responsible for the overall co-ordination and harmonisation of all procurement activities), the purchasing contractor (responsible for implementing and monitoring the procurement process at its level), and the vendor/supplier (the provider of the product).

## 4.2. Reuse program management

The reuse program management process is meant to plan, establish, manage, control, and monitor an organisation's reuse program. The entity employing and performing this organisational process shall implement and support the practice of systematic reuse in the organisation in accordance with other project lifecycle processes. Successful implementation of systematic reuse at the organisation level requires careful planning and proper management. The main outputs of this process are the organization's reuse strategy, the organization's systematic reuse capability assessment, and a set of feedback, communication and notification mechanisms

The main steps of the reuse program management are illustrated in Figure 4.

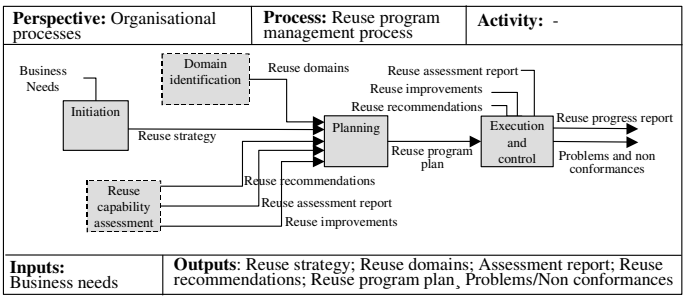


**Figure 4.** Reuse program management process

The reuse program management process starts with some initiation activities, which are driven by the organisation's business needs. After the reuse strategy is defined, the identification of the domains and the assessment of the organisations' capabilities are optional activities. The planning activities are performed considering inputs like the reuse strategy, and results from the capability assessment and domain identification (if performed). After the plan is defined, the reuse program is implemented, managed and controlled. A reuse capability assessment (including not only the reuse process, but also personnel, organisational abilities, etc.) may be performed at any time, and its results used to control the program execution in relation to the performance of any recommendation and improvement.

## 4.3. Asset management

Regardless of their overall quality and potential for reuse, assets have little value to an organization unless potential re-users know of their existence and can easily locate and understand them.

Asset management is the process of applying administrative and technical procedures throughout the life of an asset so as to: (i) identify, define, certify, classify, and baseline the asset; (ii) track modifications, migrations, and versions of the asset; (iii) record and report the status of the asset; and (iv) establish and control storage and handling of the asset, delivery of the asset to its re-users, and retirement of the asset.

The main steps of the asset management process are illustrated in Figure 5.

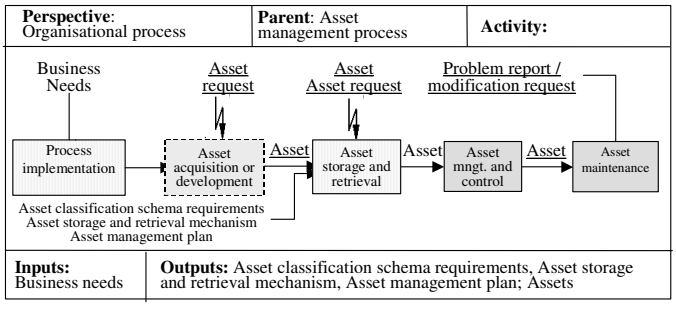


**Figure 5.** Asset management process

The activities start with the process implementation, which consists of the planning of the asset management process in the organisation. This planning activity is based on the business needs. The assets acquisition activity is driven by asset requests. The storage and retrieval of assets is respectively driven by incoming assets and asset requests, and is performed on the basis of the asset classification schema, the asset storage and retrieval mechanism and the asset management plan. The asset management and control activity is performed by the asset management system, while the asset maintenance activity is performed on the basis of the arrival of problem reports and modification requests from re-users.

### 4.4. Roles

The following roles are identified within the PDS reuse processes, where the customer could play several of them, depending on the European Space Agency and the certification authority requirements:

- *Asset manager*. The asset manager is responsible for all asset (PDS) management activities in case there is an asset management process within an organisation or project. The asset manager party may be an individual or a group of people. The customer (e.g., ESA) could play this role. Other related roles are: asset librarian, maintainer, etc.
- *Reuse program administrator*. The reuse program administrator has the responsibility to define and monitor the overall reuse program within an organisation or a big project with a centralised reuse program. The European Space Agency, the prime contractor or a top-level supplier could assume this role.
- *Reusers*. This role corresponds to those reusing a PDS. Reusers interact with the asset manager for searching assets, selecting assets, etc. The customers and the suppliers can play this role.
- *Operator*. The operator is the responsible for the operation activities, which can be partitioned into operations preparation, training, system validation, operation execution, disposal, and post-operation activities. In case there is an asset management process in the organisation or project, the operator interacts with the asset manager for the search, selection, usage and feedback of operation plan templates and procedures.
- *Maintainer*. The project appoints a maintainer to correct software errors, improve software products, migrate software to different operating environments, and retire software products. The maintainer passes any problem/non-conformance to the asset manager (when the asset management process is in place). The latter evaluates the request and modifies assets accordingly, sending a new version to the maintainer. When the maintainer needs to migrate or retire assets, it also notifies the asset manager, who will migrate or retire them. In case there is no asset management process in place, the maintainer maintains assets by applying the maintenance process.

## 5. Conclusions and future work

The PA-PDS study has shown that careful reuse of assets has the potential to lower development costs and shorten development cycles, while fulfilling the stringent dependability and safety requirements. The main results of the study are a set of product assurance requirements to support the reuse of pre-developed software (PDS) of different types (e.g., critical open source software, tools with no visibility of the source code, etc.), together with a reuse software process assessment specifying the definition of requirements and guidance material for each reuse type.

The PA-PDS study has made significant contributions regarding the different aspects to be considered for the reuse of PDS. Other aspects were left out of scope of the study, such as legal issues and domain engineering. These issues definitely need to be addressed in the future if the European Space Agency and industry aims at obtaining a significant benefit from the commercial exploitation and implementation of PDS reuse in the space domain.

As we already stated, from a subcontractor viewpoint, although reuse can improve productivity and quality, there might be some resistance to implement reuse for several reasons: (i) reuse currently entails an extra effort to 'prove' the product meets the project requirements (both domain and product assurance aspects), (ii) reuse means any customer has an expectation to pay less for the work, (iii) investment is restricted due to project-by-project funding and cost pressures, (iv) significant productivity and quality gains are only achieved when items are used several times for space project software (at least 3 - 4 times when systematic reuse is occurring). Indeed, the implementation of systematic reuse, which is necessary for companies to obtain significant benefits from reuse in the long term, is not an easy task to put into practice. That is the reason why customers should promote systematic reuse with the main aim of helping industry from moving from occasional or informal reuse to systematic reuse. This matter of implementation of reuse can be divided into three steps: (i) study of current reuse practices in industry, (ii) collection of available reusable information (e.g., quality metrics, candidate assets, etc.), and (iii) establishment of a roadmap on how to progress from occasional reuse to systematic reuse. These ideas set the basis for future projects and studies. Further ideas for future studies are summarised hereafter:

- Application of the PA-PDS results in the framework of real space projects being undertaken by the European space industry. It will complement the pilot projects [9][10].
- Development of a catalogue of products that may be suitable for reuse in space industry. This catalogue would be practically supported by an asset repository, centralising reusable assets for the space domain.
- A study covering domain engineering for space. Domain engineering aspects are crucial to maximise the benefit of both development 'with' reuse and development 'for' reuse. Indeed, the products issued from development for reuse projects will be the most suitable assets to be included in a centralised asset repository.
- Study of the legal and commercial aspects involving reuse. These aspects shall clearly cover the intellectual property, usage, payment rights, and exportation issues.
- Improvement of the reuse process capability assessment model and method. It would encompass the actions needed to obtain profit from systematic reuse. The results would include specific processes, practices, work products, and S4S capability levels needed for systematic reuse.

## References


[1] Han Van Loon, Robert Dietze, Fernando Aldea-Montero, "Software Reuse and SPICE for Space", Proc. of SPICE 2003 - Joint ESA - 3rd International SPICE Conference on Process Assessment and Improvement, 17-21 March 2003, ESTEC, Noordwijk, The Netherlands.

[2] ECSS Secretariat, "ECSS-E-40B, Space Engineering, Software", ESA-ESTEC Requirements & Standards Division, Noordwijk, The Netherlands, February 2002 (http://www.ecss.nl/).

[3] PA-PDS, "TN1 Characterisation of PreDeveloped Software Reuse and Definition of Reuse Types", Issue 3, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 19.09.2003.

[4] PA-PDS, "TN2 Survey and analysis of standards, PA Support to Pre-Developed Software", ESTEC Contract Number: 15808/01/NL/WK, Issue 1, 21.06.2003.

[5] PA-PDS, "TN3.1 Definition of software reusability assessment process", Issue 2, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 17.09.2003.

[6] PA-PDS, "TN3.2 Definition of PA requirements for different reuse types", Issue 2, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 19.09.2003.

[7] PA-PDS, "TN3.3 Technical Specification for tool(s) supporting the asset management process", Issue 1, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 24.07.2003.

[8] PA-PDS, "List of Change Requests from TN3.1 and TN3.2", Issue 2, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 12.12.2003.

[9] PA-PDS, "TN4 SCOS 2000 Evaluation report", Issue 1, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 19.09.2003.

[10] PA-PDS, "TN5 OBOSS-II Evaluation Report", Issue 2, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 19.09.2003.

[11] PA-PDS, "Training for technical personnel", Issue 1, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 17.11.2003.

[12] PA-PDS, "Training for managers", Issue 1, PA Support to Pre-Developed Software, ESTEC Contract Number: 15808/01/NL/WK, 17.11.2003.

[13] SPEC TN3, "Space Domain Specific Software Product Quality Models, Requirements and Related Evaluation Methods, SPEC/TN3", Issue 3.4, 20.2.2002.

[14] European Space Agency PASCON/WO6 CCN4/TN7: ISO/IEC TR 15504 Conformant Method for the Assessment of Space Software Processes. C. Völcker, A. Cass, Technical Note No. 7, Issue 1.0, Draft C, 19.02.00.

[15] http://descanso.jpl.nasa.gov/RCSGSO/Proceedings/Paper/A0009Paper.pdf

[16] http://www.rtems.org/

[17] http://polaris.dit.upm.es/~ork/

[18] http://spd-web.terma.com/Projects/OBOSS/Home_Page/

[19] ISO/IEC 12207:1995/Amd 1:2002, Standard for Information technology – Software life cycle processes (Amendment 1)

[20] IEEE 1517-1999, Standard for Information Technology – Software Life Cycle Processes - Reuse Processes