The ESA Software Engineering Standard and Its Applicability to HEP

Jonathan Fairclough* and Wayne Salter**
Anite Systems
*Genesis Business Park, Albert Drive, Woking, Surrey, GU21 5RW, UK
**DAS House, Quayside, Temple Back, Bristol, BS1 6NH, UK

Abstract

The European Space Agency (ESA) began development of its Software Engineering Standard PSS-05 in 1984 and this effort culminated with the release of the current version, version 2, in 1991. The standard has since been successfully applied to a large number of ESA projects. In 1994 the standard was brought into the public domain through its publication by Prentice Hall and since then has been adopted by a number of other organisations. For many of these organisations it now forms the basis of their Quality Management Systems (QMS). In 1994 CERN became interested in PSS-05 and has since that date applied it to a number of projects with varying degrees of success. In 1996 a Software Engineering Standard User Group (SESUG) was set-up by representatives of 12 European organisations; all major users of PSS-05. The SESUG has since initiated the preparation of a new version of the standard that is intended to take into account not only new developments in information technology but also the extensive experience gained by members of the SESUG through the application of PSS-05 in their domains.

The paper is intended to address the applicability of PSS-05 in the High Energy Physics (HEP) community. It starts by giving a brief overview of the standard itself and of the goals and benefits of applying it to software development projects. It summarises how the standard has been applied to projects at CERN and highlights the benefits and difficulties that have encountered on these projects. The paper then provides an overview of the improvements that are proposed for the next issue of the standard and indicates how these address the problem areas experience by CERN applying version 2 of the standard.

1 PSS-05

Projects can succeed or fail. Projects aim to deliver on time and within budget, and project management is the technique for achieving this. Similarly projects aim to deliver products that are fit for purpose, and therefore all projects need some form of quality management. ESA recognised some time ago that the absence of project and quality management is a major risk to any project, and developed a comprehensive definition of how to manage a software project and as a result develop quality software. This definition is contained in the PSS-05 series of standards and guides.

The PSS-05 standard applies to all software produced for ESA and includes a life cycle reference model with defined activities, inputs and outputs. Moreover it describes all associated management activities - project management, configuration management, quality assurance, and verification and validation - that are required to be performed to ensure a successful outcome to the overall project.

To ease the production of the major documents and to monitor the progress of the project throughout the software life cycle, the PSS-05 standard also contains a number of document templates, as well as clearly identifying when major reviews, including both the customer and the software developer, should take place.

The PSS-05 guides provide an easy to understand set of guidelines covering all aspects of a software development project, and following the standard leads to discipline and quality. Newcomers to the PSS-05 standard can be secure in the knowledge that it has been tried and tested in a large number of projects, and that the experience gained is being channelled back into evolving the standard through the SESUG [1], of which CERN is an active member.

The PSS-05 standard has been published as ″Software Engineering Standards″ [2]. The ten associated guides have been collected into a book and published as ″Software Engineering Guides″ [3].

The structure of the PSS-05 standard is based around a software life cycle reference model that defines the activities for each phase as well as identifying the inputs and outputs to/from each phase. The following phases are defined:

- User Requirements (UR) or *′problem definition'* phase. Requirements to be written from a user′s point of view. The system is treated as a ″black box″.
- Software Requirements (SR) or *′problem analysis¢* phase. The user′s requirements are analysed and a set of software requirements produced. Requirements on the internal functioning of the black box are defined to meet the user requirements.
- Architectural Design (AD) or *′solution¢* phase. A set of software components and their interfaces are defined as a framework for developing the software.
- Detailed Design (DD) or *′implementation¢* phase. The design outlined in the AD phase is defined in more detail and the software is coded, tested and documented.
- Transfer (TR) or *′handover'* phase. The software is installed in the operational environment and the capabilities of the software, as given in the User Requirements Document (URD), are demonstrated to the user.

- Operation and Maintenance (OM) phase. The software enters practical use and is maintained.

The PSS-05 standard includes three applications of the reference model: waterfall, incremental delivery, evolutionary delivery. The waterfall assumes that all the above phases are performed in a sequential manner. The incremental delivery approach follows the waterfall approach up to the AD phase and then allows the DD, TR and OM phases to be split into multiple phases. The evolutionary approach allows the complete life cycle to repeated many times. The PSS-05 guides define a process model for each phase including the corresponding life cycle management activity. They also provide a mapping to software engineering methods and tools.

The major strengths of the PSS-05 series are they:

- define the software engineering discipline
- can be used as the basis of ISO 9001 standard QMS
- efficiently incorporate products and processes in a life cycle model
- easy to understand
- results in quality software
- tried and trusted
- training support available

2 Software projects at CERN

There are two types of project, those under the full responsibility of CERN, and those under the responsibility of the experiment collaborations. CERN projects are easier as they are performed in one place, are under the full responsibility of CERN, including the management of manpower and budget resources, and are performed fully under the CERN management hierarchy. CERN projects have the following characteristics:

- Users are often developers
- Involvement of highly qualified physicists.

Such projects are generally performed by a team of experienced staff working in one group within CERN. On the other hand, projects performed in the experiment collaborations are far more complicated, and have the following characteristics in addition to those of the CERN projects:

- Large distributed collaborations
- Limited management hierarchy
- Lots of Ph.D. students to do programming
- Experiments and therefore changing needs
- Developers are often not around for maintenance
- Developers are often only around for a few years

The last two points also apply to CERN in-house projects but to a lesser extent. These characteristics make the risks below very likely:

- communication problems
- requirements change
- personnel change
- technical novelty
- cost underestimation

A structured approach to project and quality management is essential for success according to all the criteria below:

- within budget
- within timescale
- meaningful test results
- scientific discoveries

Often success on one of the criteria is used to excuse failures on other criteria. Scientists are both users and engineers in CERN projects. We believe that scientists must learn and apply engineering techniques if large CERN projects are to be successful. In short, scientists must learn to switch roles. Furthermore, we believe that the situation at CERN is representative of the HEP community at large.

3 CERN experience with PSS-05

Many people at CERN have recognised that to ensure the success of its software projects the application of a proper software engineering standard is essential. PSS-05 was therefore adopted by CERN and has been applied on a large number of projects (> 20) in a wide range of technical areas.

It should be noted that although PSS-05 has been applied to a large number of projects, it has generally only be selectively applied because of a number of reasons which are discussed below. That is to say that it has typically only been applied for specific phases of the life cycle, and predominantly for the user requirements phase.

Below we look at the two main areas in which PSS-05 has been applied and the experience gained through it; In-house software Development and Information System (IS) Acquisition.

To aid the usage of PSS-05 at CERN, and in particular in the production of URDs, a URD template has been produced using Framemaker, which can then be converted to a Web browser readable format using a CERN developed tool called Webmaker. Similarly Word 7.0 templates have been produced for all PSS-05 defined documents. Furthermore a number of training courses have been conducted to introduce PPS-05 to CERN and the experiment teams working there and additionally some specific consultancy has been taken to assist in its application to a number of projects.

3.1 In-house software development projects

3.1.1 Usage

PSS-05 has been applied to a number of software development projects, particularly for software systems for the Large Hadron Collider (LHC) experiments, for both on-line and off-line software. The motivation to use PSS-05 in this area has been to encourage good software engineering practices and to develop a standard way of working across the large number of institutes participating in the LHC experiment collaborations. To-date PSS-05 has primarily been used by users as a means of clearly specifying and documenting their requirements, which are then used as the basis against which the systems will be developed. On a much smaller number of projects it has further been used as a software development methodology throughout the complete development cycle. For all these

projects the clear definition of activities to be performed, deliverables to be produced and reviews to be held as defined by PSS-05 has aided the overall development process.

3.1.2 Benefits

As stated above PSS-05 has been extensively applied for the requirements definition phase of numerous projects and has been shown to provide significant benefits, both to the developers in having a defined set of requirements from which to work, but also to the users in helping them to define their needs. Major benefits are as follows:

- the users are forced to think clearly about their problem at an early stage
- it produces a clear specification of what is required
- it leads to a common understanding for all involved
- allows commonality between similar systems to be more clearly identified
- document against which acceptance testing can be done
- allows traceability through design and development
- a thorough URD helps to reduce the scope of the changes later - but cannot eliminate them altogether

Few projects have gone past this phase, partly due to the early nature of the LHC developments, and partly due to the problems described below. However, for the projects that have applied the standard right through the complete development cycle it has been shown to be effective in encouraging good engineering practices and a consistent way of working across a large number of participating sites. For these projects the standard was not applied as is, but rather a tailored version was applied.

3.1.3 Problem areas

First of all there appears to be a cultural problem in introducing standardisation of any kind into the HEP community and this applies equally well to the introduction of software engineering standards. However, for those who have tried to apply PSS-05 a number of real issues were also raised:

- It is perceived as too rigid and bureaucratic by the majority of users
- It cannot generally be applied as is, and there are no guidelines provided on how to tailor it to the specific needs of the project
- Although it contains three life cycle models it is largely based around the waterfall life cycle which is generally not applicable for HEP projects
- It does not easily lends itself to Object Oriented (OO) projects
- Is purely a software engineering standard and does not include guidelines for H/W and software projects

The perception of rigidity and formality can be removed by providing training. Most new users of the standards need to be educated in the concepts underlying the practices before the practices can be accepted. Likewise, training can also educate users of the standard in applying it to their projects.

The reference model that is used within PSS-05 to describe the logic of software development corresponds directly to the waterfall model, but they are not necessarily one and the same. Unfortunately this difference is often not understood, and many people new to the PSS-05 standard follow a waterfall approach by directly applying the reference model without considering the needs of their project. Project managers should design a project-specific life cycle approach based upon one of the approaches in the PSS-05 standard.

ESA has successfully used OO methods in numerous projects where PSS-05 has been applied. The lessons learnt include:

- the mapping of OO methods to the life cycle
- the start and end criteria for each phase
- the importance of ′use cases′ for coordinating all phases of development (the use cases replace the functional requirements in the development methodology)

ESA is currently preparing a guide for OO projects based upon the experience above.

With regard to system projects, generalisations of PSS-05 for system projects have been developed by some members of the SESUG and these ideas and the experienced gained applying them will be incorporated into the follow-on version, see section 4.

3.2 IS acquisition

3.2.1 Usage

In a number of cases PSS-05 has been used to aid the control of IS Acquisition (outsourcing projects). In these cases a specific development has been performed by an external contractor to meet the needs of CERN. In particular PSS-05 was applied to help the definition of the work specification through the production of detailed URDs, and to manage the contract using the guidelines contained in PSS-05 for the software project management. PSS-05 has additionally been used to produce specifications, against which off-the-shelf products are evaluated for suitability for a particular need.

3.2.2 Benefits

The production of a specification according to the guidelines of a URD significantly helped to define clearly the scope of the IS development, and formed a good basis for the contractual agreement between CERN and the contractor. In defining PSS-05 as the software engineering methodology to be followed by the contractor, the CERN personnel monitoring the contract were given a clear baseline against which to monitor the progress of the development. A risk seen by CERN in outsourcing software development has been the potential need to rely on the contractor for maintenance and further modifications to the system due to a lack of in-depth knowledge by the CERN staff of the system. However, it has been shown that in holding the formal technical reviews as defined in PSS-05 and by insisting on the deliverables specified in PSS-05, the CERN staff have been able to follow closely the development and were able

to take over the maintenance of the system and subsequently perform modifications to it.

3.2.3 Problem areas

Although the use of PSS-05 has produced significant benefits it does not contain guidelines for the management of the tender process or specific guidelines for contract management.

3.3 Summary

The experience seen at CERN and within the LHC collaborations is assumed to be fairly typical of HEP. There seems to be a certain reticence on the part of physicists to work in a formal manner as this is typically perceived to stifle creativity and reduce flexibility. Furthermore, PSS-05 is perceived as being very document heavy and therefore time consuming to apply. On the face of it there is a certain amount of truth in this. However a number of projects have shown that with a flexible approach and some customising of the standard, it can be applied in a way that reduces the effort to an acceptable level, and still produces noticeable benefits. The main benefit to-date has been shown in the area of requirements definition which is then documented in the URD.

In the HEP environment where the development team is often not around for the maintenance, the development of documentation/standard approach/quality helps future maintenance. Well documented and engineered software systems will significantly improve the maintenance of the systems over the projected long life-times. This is especially important when taken together with reduced budgets and manpower and with an operational lifetime of more than 15 years as is the case for LHC.

4 Developments with version 3

At a meeting of the SESUG in March of this year it was decided to produce an update to PSS-05 to reflect the changes in Information Technology (IT) and to take account of the considerable experience already gained with the current version PSS-05 [1]. At this meeting, a suggested list of changes, which had been derived from suggestions made by the members of the SESUG, was presented in draft form. Version three is expected to be produced in 1998. The major issues which will be addressed in the new version are described below.

4.1 Compliance with ISO 9001/9000-3

As stated previously, many of the organisations which form the SESUG have based their QMS on PSS-05. Many of these organisations have, or intend, to apply for ISO 9000 certification. Therefore, it is clearly desirable that the follow-on version of PSS-05 should be fully compliant with the relevant ISO standards.

4.2 Compliance with ISO/IEC 12207

This new international standard defines the vocabulary of software engineering, and defines at the top level that software engineering consists of a number of processes. These are:

- primary life cycle processes, covering acquisition, supply, development, operations and maintenance
- supporting processes, such as configuration management, verification, validation and quality assurance
- organisational processes, such as management, infrastructure and improvement

Any new ′organisational′ standard, such as a follow-on to PSS-05, needs to comply with ISO/IEC 12207 by using its vocabulary and process definitions which should then be tailored to the needs of the organisation.

4.3 Consideration of small projects

PSS-05 was produced in an environment where large projects requiring hundreds of man-months of effort were common place. The standard was therefore optimised for the management of large projects. However, PSS-05 is intended to be used on all types of software projects, large or small, and for any software application. Experience has should that there are always many more small software projects as large ones, but people applying the standard on small projects have been forced to modify PSS-05 in several ways to be able apply it for such projects. This experience is being incorporated into the new version which will primarily look at smaller projects, but indicate additional practices that should be applied to larger projects. See also the comments regarding tailoring below.

4.4 Consideration of OO development

OO has caused a paradigm shift in the way software engineers analyse and design software. Programming languages have changed from procedural languages such as Fortran and C to OO languages such as C++, Ada and Eiffel. The follow-on to PSS-05 needs to be consistent with OO analysis, design and development methods and reflect the increasing dominance of OO in software engineering.

4.5 Consideration of contract/tendering issues

Contracting and tendering issues were intentionally left out of PSS-05 because ESA already had a separate set of standards/procedures for handling these issues. However, PSS-05 has now been adopted by a number of other organisation for which it also forms the basis of their QMS. It is essential that contract and tendering issues are now addressed in the follow-on version.

4.6 Simplification of document templates

The document templates were again based on the concept of large software developments and have proved to be too heavy for smaller projects. The follow-on will provide simplified document templates to reduce the load for small projects, which is in-line with point 3 above.

4.7 Software re-use and COTS products

One way to reduce ′time-to-market′ and deliver better systems faster is to re-use software. This may be done by buying commercial off-the-shelf (COTS) software, obtaining ′shareware′ or re-using software already

developed by the organisation. The follow-on to PSS-05 must deal with the issues involved with the use of such software. There are major quality issues involved with re-use and it will certainly help to include aspects of the ISO 9001 purchasing requirements.

4.8 Guidance for projects that are not pure software

Most modern projects are not simple software projects, but are rather development projects′ where both hardware and software aspects are involved. As described earlier PSS-05 was developed as a software engineering standard, but it is recognised that it would be beneficial if the standard addressed complete IT projects.

4.9 More life cycle approaches

PSS-05 currently contains three life cycle models. However, it is clear that in practice these life cycles are not sufficient to covering all projects' needs. The follow-on of PSS-05 will emphasis processes, activities and tasks that are fundamental, and stress that the life cycle should be adapted or designed individually to meet the needs of the project.

4.10 Tailoring

The experience has shown that for the most efficient usage of the standard it is important to tailor it to the specific needs of the project. The current version does not include any guidelines on how to perform this. The follow-on version will therefore contain a number of guidelines on how to tailor PSS-05 to the particular needs of the project concerned.

Given the list of proposed changes one might be tempted to ask why one shouldn′t throw away PSS-05 and start from scratch. However, the users of PSS-05 have made considerable investments in developing a successful software engineering culture around the standard, and it is essential that the follow-on builds upon those investments. As such the follow-on should be seen as being an evolution of the current version, which maintains the general philosophy and style of it while incorporating the changes which are clearly necessary to enable PSS-05 to remain a popular and applicable software engineering standard for modern IT projects.

5 Conclusions

PSS-05 has now been applied at CERN on a large number of projects despite an initial reluctance from the HEP community. Where it has been applied a number of problem areas have been identified, but nonetheless significant benefits have also been seen. A follow-on version has been designed that addresses the majority of the problems highlighted by the CERN usage and will enable PSS-05 to be applied in a more straightforward manner. This initial experience shows that PSS-05 can be successfully used in the HEP community once the initial reticence of applying formal methods has been overcome, and if some flexibility is allowed. The follow-on version should provide an even better basis for good software engineering practices to be applied to HEP projects. It is important to consider that future HEP projects are likely to be done on lower budgets and with less resources, and as such a more efficient approach to software development will be necessary, particularly to reduce the amount of effort required for maintenance and to avoid a re-engineering of the system as a result of turn over of staff and lack of documentation.

References

[1] Proceedings of the SESUG Workshop, CERN, March 6th-7th March 1997, available from Jon Fairclough at Anite Systems

[2] C. Mazza, J. Fairclough, B. Melton, D de Pablo, A. Scheffer, R. Stevens, Prentice-Hall, 1994. ISBN 131 065 688

[3] Software Engineering Guides, C. Mazza, J. Fairclough, B. Melton, D de Pablo, A. Scheffer, R. Stevens, M. Jones, G. Alvisi, Prentice-Hall, 1995. ISBN 0-13-449281-1.