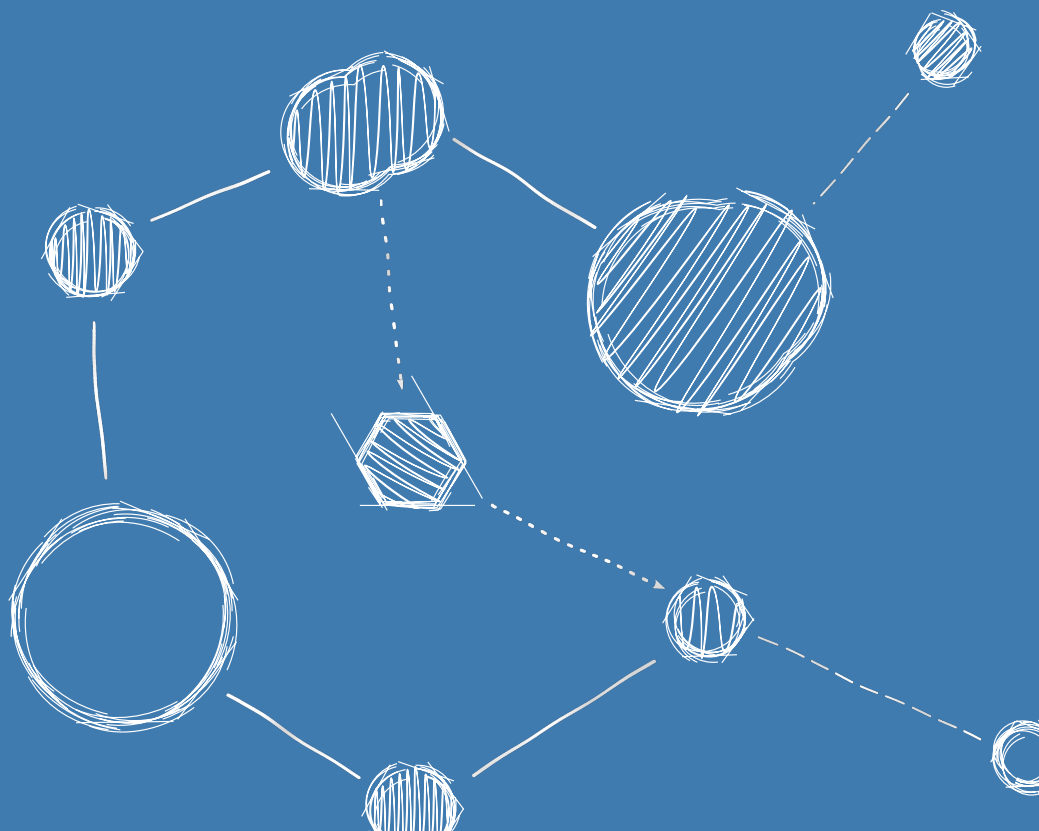


LEAN OS – AN OPERATING SYSTEM FOR THE SSDP

Product Quality Assurance Plan



Product Quality Assurance Plan

Reference: LEANOS-UVIE-PAQ-001

Version: Issue 1.0, June 01, 2017

Prepared by: Armin Luntzer¹

Checked by: Roland Ottensamer¹, Christian Reimers¹

Approved by: Franz Kerschbaum¹

¹ Department of Astrophysics, University of Vienna

Copyright ©2017

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Front-Cover, no Logos of the University of Vienna.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 4 |
| 1.1 | Purpose of the Document | 4 |
| 2 | Applicable and Reference Documents | 5 |
| 3 | Terms, Definitions and Abbreviated Items | 6 |
| 3.1 | Acronyms | 6 |
| 3.2 | Glossary | 7 |
| 4 | System Overview | 10 |
| 5 | Software Product Assurance Programme Implementation | 11 |
| 5.1 | Organisation | 11 |
| 5.2 | Responsibilities | 12 |
| 5.3 | Resources | 12 |
| 5.4 | Reporting | 13 |
| 5.5 | Quality models | 13 |
| 5.6 | Risk Management | 14 |
| 5.7 | Supplier selection and control | 14 |
| 5.8 | Methods and tools | 14 |
| 5.9 | Process assessment and improvement | 15 |
| 5.10 | Operations and maintenance | 15 |
| 6 | Software Process Assurance | 16 |
| 6.1 | Software development cycle | 16 |
| 6.2 | Project plans | 17 |
| 6.3 | Software dependability and safety | 17 |
| 6.4 | Software documentation and configuration management | 17 |
| 6.5 | Process metrics | 18 |
| 6.6 | Reuse of software | 18 |
| 6.7 | Product assurance planning for individual processes and activities | 18 |
| 6.8 | Procedures and standards | 18 |
| 7 | Software Product Quality Assurance | 19 |
| 8 | Compliance Matrix to Software Product Assurance Requirements | 20 |

Revision History

| Revision | Date | Author(s) | Description |
|----------|------------|-----------|-------------------|
| 0.1 | 29.05.2015 | AL | initial version |
| 1.0 | 01.06.2017 | FK | document approved |

1. Introduction

1.1 Purpose of the Document

This document describes the [Software Quality Assurance \(SQA\)](#) activities that will be performed in the development of the the operating system kernel LeanOS. The document is applicable for all versions of LeanOS and shall be updated as needed.

LeanOS targets the [Scalable Sensor Data Processor \(SSDP\)](#), and to a lesser extent, its compatible predecessor, the [Massively Parallel Processor Breadboarding system \(MPPB\)](#) v2.x [7]. It is intended to be used in unmanned space applications of (at least) Software Criticality Level C.

This document follows the document structure for software product assurance plans found in Annex B of ECSS-Q-ST-80C [2].

2. Applicable and Reference Documents

- [1] *ECSS-Q-ST-10-09C Space product assurance - Nonconformance control system*. ESA Requirements and Standards Division, 2009.
- [2] *ECSS-Q-ST-80C Space product assurance - Software product assurance*. ESA Requirements and Standards Division, 2009.
- [3] *LeanOS Software Requirements Specification*. 2017.
- [4] *LeanOS Test Plan*. 2017.
- [5] *LeanOS Test Specification*. 2017.
- [6] Armin Luntzer et al. *A Lightweight Operating System for the SSDP*. 2016. URL: <https://indico.esa.int/indico/event/102/session/28/contribution/26>.
- [7] *Massively Parallel Processor Breadboarding Datasheet*. 2016.

3. Terms, Definitions and Abbreviated Items

3.1 Acronyms

| | |
|--------------|--|
| API | Application Programming Interface |
| CPU | Central Processing Unit |
| DMA | Direct Memory Access |
| DRD | Document Requirements Definition 17 |
| DSP | Digital Signal Processor |
| FDIR | Fault Detection, Isolation and Recovery 17 |
| FSW | Flight Software 11 |
| ILP | Instruction Level Parallelism |
| MPPB | Massively Parallel Processor Breadboarding system |
| NCR | Non-Conformance Reports 12, 13, 17, 18 |
| NGAPP | Next Generation Astronomy Processing Platform |
| NoC | Network On Chip |
| NRB | Nonconformance Review Board 17 |
| PA | Product Assurance 11 |
| PAM | Product Assurance Manager 11, 12, 14, 17 |
| RAM | Random-Access Memory |
| RID | Review Item Discrepancy 18 |
| RISC | Reduced Instruction Set Computing |
| RSA | RUAG Space Austria 10 |
| SDD | Software Design Document 13 |
| SMP | Symmetric Multiprocessing |
| SoC | System On Chip |
| SPAMR | Software Product Assurance Milestone Report 13, 18 |
| SPR | Software Problems Reports 12, 13, 18 |
| SQ | Software Quality 11, 12 |
| SQA | Software Quality Assurance 4, 11, 12, 18 |
| SQAM | Software Quality Assurance Manager 11, 18 |
| SRS | Software Requirement Specification 13 |
| SSDP | Scalable Sensor Data Processor |
| SSS | System Software Specification 13 |

TP Test Plan [14, 17](#)
TS Test Specification [14](#)
UVIE University of Vienna [10](#)
VLIW [Very Long Instruction Word](#)

3.2 Glossary

Application Programming Interface (**API**)

The Application Programming Interface defines how a developer can write a program that requests services from an operating system or application. [APIs](#) are implemented by function calls composed of verbs and nouns, i.e. a function to execute on an object.

Central Processing Unit (**CPU**)

The Central Processing Unit is the electronic circuitry that interprets instructions of a computer program and performs control logic, arithmetic, and input/output operations specified by the instructions. It maintains high-level control of peripheral components, such as memory and other devices.

Digital Signal Processor (**DSP**)

A Digital Signal Processor is a specialised processor with its architecture targeting the operational needs of digital signal processing.

Direct Memory Access (**DMA**)

Direct Memory Access is a feature of a computer system that allows hardware subsystems to access main system [Random-Access Memory \(Random-Access Memory \(RAM\)\)](#) directly, thereby bypassing the [Central Processing Unit \(CPU\)](#).

Instruction Level Parallelism (**ILP**)

Instruction-level parallelism (ILP) is a measure of how many instructions in a computer program can be executed simultaneously by the [CPU](#).

LEON2

The LEON2 is a synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. It is highly configurable and particularly suitable for [System On Chip \(SoC\)](#) designs. Its source code is available under the GNU LGPL license

LEON3

The LEON3 is an updated version of the [LEON2](#), changes include [Symmetric Multiprocessing \(Symmetric Multiprocessing \(SMP\)\)](#) support and a deeper instruction pipeline

LEON3-FT

The LEON3-FT is a fault-tolerant version of the [LEON3](#). Changes to the base version include autonomous error handling, cache locking and different cache replacement strategies.

Massively Parallel Processor Breadboarding system (MPPB)

The Massively Parallel Processor Breadboarding system is a proof-of-concept design for a space-hardened, fault-tolerant multi-DSP system with various subsystems to build a powerful digital signal processing system with a high data throughput. Its distinguishing features are the [Network On Chip \(Network On Chip \(NoC\)\)](#) and the [Xentium DSPs](#) controlled by a [LEON2](#) processor. It was developed under ESA contract 21986 by Recore Systems B.V.

Network On Chip (NoC)

A Network On Chip is a communication system on an integrated circuit that applies (packet based) networking to on-chip communication. It offers improvements over more conventional bus interconnects and is more scalable and power efficient in complex [System On Chip \(SoC\)](#) designs.

Next Generation Astronomy Processing Platform (NGAPP)

Next Generation Astronomy Processing Platform was an evaluation of the [MPPB](#) performed in a joint effort of RUAG Space Austria and the Department of Astrophysics of the University of Vienna. The project was funded under ESA contract 40000107815/13/NL/EL/f.

Random-Access Memory (RAM)

Random-Access Memory is a type of memory where each memory cell may be accessed directly via their memory addresses.

Reduced Instruction Set Computing (RISC)

RISC is a [CPU](#) design strategy that intends to improve performance by combining a simplified instruction set with a microprocessor architecture that is capable of executing an instruction in a smaller number of clock cycles.

Scalable Sensor Data Processor (SSDP)

The Scalable Sensor Data Processor (SSDP) is a next generation on-board data processing mixed-signal ASIC, envisaged to be used in future scientific payloads requiring high-performance on-board processing capabilities. It is built upon a heterogeneous multicore architecture, combining two [Xentium DSP](#) cores with a general-purpose [LEON3-FT](#) control processor in a [Network On Chip \(NoC\)](#).

SPARC

SPARC ("scalable processor architecture") is a [Reduced Instruction Set Computing \(RISC\)](#) instruction set architecture developed by Sun Microsystems in the 1980s. The distinct feature of SPARC processors is the high number of [Central Processing Unit \(CPU\)](#) registers that are accessed similarly to stack variables via "sliding windows".

Symmetric Multiprocessing (SMP)

Symmetric Multiprocessing denotes computer architectures, where two or more identical processors are connected to the same periphery and are controlled by the same operating system instance.

System On Chip (SoC)

A System On Chip is an integrated circuit that combines all components of a computer or other electronic system into a single chip.

Very Long Instruction Word (VLIW)

Very Long Instruction Word is a processor architecture design concept that exploits [Instruction Level Parallelism \(ILP\)](#). This approach allows higher performance at a smaller silicon footprint compared to serialised instruction processors, as no instruction re-ordering logic to exploit superscalar capabilities of the processor must be integrated on the chip, but requires either code to be tuned manually or a very sophisticated compiler to exploit the full potential of the processor.

Xentium

The Xentium is a high performance [Very Long Instruction Word \(VLIW\) DSP](#) core. It operates 10 parallel execution slots supporting 32/40 bit scalar and two 16-bit element vector operations.

4. System Overview

In the course of the [NGAPP](#) activities, an evaluation of the [MPPB](#) was performed in a joint effort of [RUAG Space Austria \(RSA\)](#) and the Department of Astrophysics of the [University of Vienna \(UVIE\)](#). While the original intent of the work of [UVIE](#) was to quantify the performance of the [Xen-tium DSPs](#) and the [MPPB](#) as a whole with regard to on-board data treatment and reduction in an astronomical mission setting, it was found that, given the highly innovative nature of this new processing platform, a novel approach was needed concerning the management of system resources, [DMA](#) mechanics and [DSP](#) program design for best efficiency and turnover rates. Consequently, [UVIE](#) developed an experimental operating system to stably drive the [DSP](#) cores and the [MPPB](#) close to its performance limit. LeanOS is a development based on this operating system concept.

Further details on LeanOS may be found in [\[6\]](#).

5. Software Product Assurance Programme Implementation

5.1 Organisation

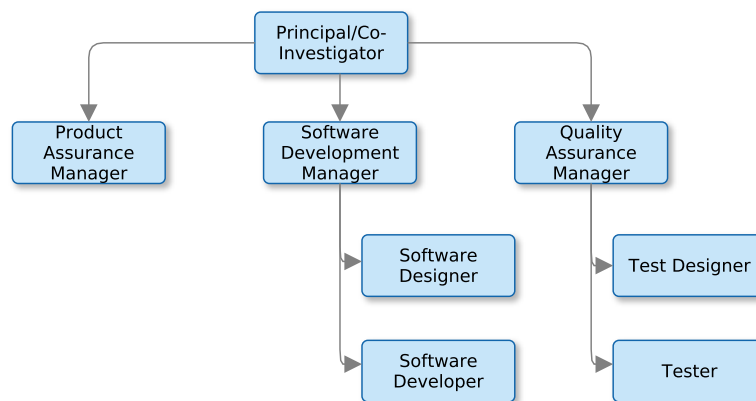


Figure 5.1: A typical organisational structure in a software development group.

The [SQA](#) organisation (Figure 5.1) for the development of LeanOS is commensurate to the scope and size of the working group dedicated to [Flight Software \(FSW\)](#) development at the Department of Astrophysics of the University of Vienna.

As projects are typically carried out by a small group of people, several of the project tasks may be executed by one and the same person.

There is no formal [Product Assurance \(PA\)](#) section at the Department of Astrophysics. Instead, the [Product Assurance Manager \(PAM\)](#), together with the other team leaders, forms a [PA](#) group for the specific project. In particular, the [PAM](#) also takes over the role of the [Software Quality Assurance Manager \(SQAM\)](#) and nominates people as [Software Quality \(SQ\)](#) personnel as needed and if applicable.

[PAM](#) and (internal) [SQ](#) personnel maintain a level of independence from the project and the software development.

5.2 Responsibilities

Formal responsibilities of the **PAM** and **SQ** personnel include process and product assessments:

- System and System Design Reviews
- Requirements Management
- Test Management
- **Non-Conformance Reports (NCR)** Management
- **Software Problems Reports (SPR)** Management
- Document Reviews and Review Packages
- Software Development Records
- Software Configuration Management

Given the organisational and personnel constraints described in the previous section, activities of the **SQ** personnel shall be conducted in an informal way between the members. The **PAM** shall assess and give assistance as to the formal aspects of the product and process assurance. Examples of this are configuration management or the state of completeness of review packages.

The **PAM** ensures the compliance with the software development processes described in the corresponding software development plan if applicable.

The main responsibilities of **SQA** shall remain with the project team, in particular the relevant software developer. In particular, this concerns the documentation of any findings from software related activities, e.g. bug tracking and repeated software testing during the development cycle.

5.3 Resources

Staffing to support software assurance activities must be balanced against the other activities that are to be carried out within the project at any given time period and against underlying availability of personnel.

The required resource levels shall be discussed, adjusted and managed in conjunction with project management to ensure maintenance of adequate support within the life cycle of the project.

5.4 Reporting

Since the project is not concerned with external partners, reporting is done informally, rather than via [Software Product Assurance Milestone Reports \(SPAMRs\)](#).

Still, these informal reports shall cover:

- [System Software Specification \(SSS\)](#) coverage
- [Software Requirement Specification \(SRS\)](#) coverage
- [Software Design Document \(SDD\)](#) coverage
- Testing activities
- Verification activities
- [NCR](#) and action status
- [SPR](#) and action status
- Product metrics
- Compliance and adherence to design and coding standards
- Methods and tooling

No minutes of meeting or other record has to be made of meetings regarding these informal reports.

5.5 Quality models

In ECSS-Q-ST-80C[2], these potential quality characteristics of a software application are identified:

- | | |
|--------------------------|--------------------------------------|
| • functionality | • security |
| • reliability | • usability |
| • maintainability | • efficiency |
| • reusability | • portability |
| • suitability for safety | • software development effectiveness |

Functionality is explicitly covered in the [SRS](#)[3]. Similarly, the [SRS](#) also covers maintainability, safety and security requirements, as well as efficiency.

Reliability is covered implicitly through the requirements formulated in the [Test Plan \(TP\)](#)[4] and [Test Specification \(TS\)](#)[5]. which are designed to demonstrate the ability of the software to satisfy its requirements within its operational domain.

Usability is not formally relevant for LeanOS, but rather implied by software design and its intended application.

Portability and reusability are implicit by the type of software product (an embedded operating system).

Software development effectiveness is not covered explicitly, but is implicitly covered by the definition of the development process.

5.6 Risk Management

If software-specific risks are identified during development, they shall be incorporated into a risk portfolio by the [PAM](#). The status of corrective and/or preventive actions shall be tracked, including a general action plan and progress per identified risk.

Given the small group of involved personnel, communication is straightforward and ensures an efficient flow of information on an informal level.

5.7 Supplier selection and control

Not applicable.

5.8 Methods and tools

The following tools are used for the management process:

git source code and document version control

TeX Live document preparation

The following tools are used for the software development and testing process:

sparc-gcc C language compiler for [SPARC](#)

xentium-clang C language compiler for the [Xentium](#)

gcov code coverage testing

doxygen in-line software and [API](#) documentation

All of the listed tools are well established and are at a high level of maturity.

5.9 Process assessment and improvement

Assessment and possible improvement of the development process shall be done internally on an informal basis.

5.10 Operations and maintenance

No special quality measures are defined for operations or maintenance.

6. Software Process Assurance

6.1 Software development cycle

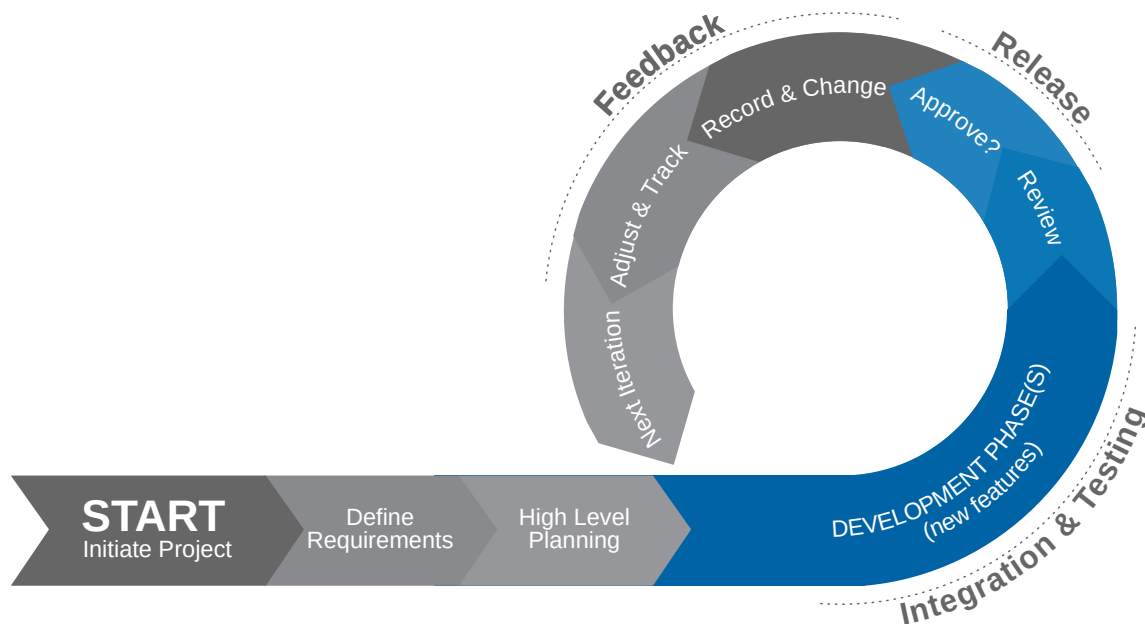


Figure 6.1: The life cycle of the agile software development model. After the initial identification of requirements and a high level description of the software product, steps of implementation of new features and testing are followed by review and approval of those features. If the results of the development are approved, a new version is released. Based on the feedback, the development target is adjusted and the cycle is started over.

For the software development life cycle, the *Agile* model (Figure 6.1) is selected to be used for the development of LeanOS. This model keeps the initial planning and analysis on a very high level that is just detailed enough to outline the scope of the project. The development process is a series of iterations, in which a feature is taken from start to finish during a single iteration and released intermediately during or at the end of an iteration. There are no distinct stages during a cycle, but activities such as analyzing, designing, developing and testing of a feature are executed in turn within each iteration.

The benefit of this approach is the freedom to quickly and flexibly react to changing circumstances such as changes to hardware features of the underlying platform, or evolving use cases for the software product. Changes can be discussed and new features can be implemented or those with shortcomings removed or replaced, based on feedback of the user base. However, since this type of work requires experienced developers that are capable of independent de-

cision making and a wide skill set that covers efficient design and implementation, as well as thorough test conception, adequate personnel must be carefully selected when staffing.

6.2 Project plans

A [Test Plan](#) is given in [\[4\]](#).

6.3 Software dependability and safety

LeanOS is intended to be used in unmanned space applications of (at least) Software Criticality Level C. However, [Fault Detection, Isolation and Recovery \(FDIR\)](#) analyses and derived measures as well as the particular configuration of the operating system to be used are highly dependent on the use case and must be defined at a different stage. Fundamental dependability on a software component level is ensured by verification and validation testing as part of the software development process.

6.4 Software documentation and configuration management

Documents are written and typeset in \LaTeX and compiled with the **TeX Live** distribution of tools. For configuration management of both documents and source code, **git** is used, which provides implicit protection of software and document versions by cryptographic means.

Document identification, content, change records and issue/revision handling shall be oriented along the applicable [Document Requirements Definition \(DRD\)](#) of the corresponding ECSS standards.

The [PAM](#) shall verify any documentation before release as to formal standards, such as document identifier, completeness of information, date, issue and revision.

As part of the preparation of a documentation package for external review, both the project manager and the [PAM](#) shall perform a full review of the contained documentation.

All released documents, unless otherwise marked, can be considered signed, even without the inclusion of an actual signature.

Nonconformance management and closure processes shall follow the procedures outlined in [\[2\]](#) and [\[1\]](#) if needed when dealing with external partners. Internally, [NCRs](#) are handled informally, no [Nonconformance Review Board \(NRB\)](#) is foreseen to be instated.

6.5 Process metrics

Actions, [NCRs](#), [SPRs](#), resulting deviations and waivers as well as [SQA](#) statistics shall be managed by the [SQAM](#) as applicable. No formal method of how records are kept is specified.

Any recorded metrics shall be made available on demand.

Metrics regarding the actual program code shall be managed by lead software developer. These metrics shall be part of test reports if applicable.

6.6 Reuse of software

If software is reused from other projects, a full qualification package shall be included to the same level of any newly developed code.

6.7 Product assurance planning for individual processes and activities

The [SQA](#) shall assure that all activities concur with the planning and the definitions given in this plan.

Compliance will be monitored informally and internally, accompanying the development of the software.

This shall include:

- review of data packages as to adequacy and completeness
- [Review Item Discrepancy \(RID\)](#) management
- [NCR](#) management
- [SPR](#) management
- action management
- provision of a [SPAMR](#) (optional, only if needed)

6.8 Procedures and standards

Procedures and standards are described in the preceeding sections of this document.

7. Software Product Quality Assurance

Product metrics of the software code include:

- number of source lines of code
- number of comments
- functional complexity (cyclomatic complexity, lines per function, ...)
- code coverage (via unit tests)
- completeness of code commentary/design description
- test coverage
- number of test failures

These metrics are collected via automated analysis scripts or manually and reported in the corresponding test report.

8. Compliance Matrix to Software Product Assurance Requirements

The compliance matrix will be made available as a separate document.