Collaborative Large-scale Integrating Project

Open Platform for EvolutioNary Certification Of
Safety-critical Systems

Process-specific service infrastructure:
Methodological Guide
D7.6

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP7: Transparent Certification and Compliance-aware Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissemination level:</td>
<td>PU (Public)</td>
</tr>
<tr>
<td>Status:</td>
<td>Draft (V0.5)</td>
</tr>
<tr>
<td>Date:</td>
<td>23 September 2014</td>
</tr>
<tr>
<td>Responsible partner:</td>
<td>Alessandra Martelli (INT)</td>
</tr>
<tr>
<td>Contact information:</td>
<td><a href="mailto:Alessandra.martelli@intecs.it">Alessandra.martelli@intecs.it</a></td>
</tr>
</tbody>
</table>

PROPRIETARY RIGHTS STATEMENT
This document contains information, which is proprietary to the OPENCOSS Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the OPENCOSS consortium.
## Contributors

<table>
<thead>
<tr>
<th>Names</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alessandra Martelli, Paolo Panaroni (authors)</td>
<td>INTECS</td>
</tr>
<tr>
<td>Andrea Musone, John Favaro (internal reviewers)</td>
<td>INTECS</td>
</tr>
<tr>
<td>Adrian Larkham (author)</td>
<td>ATEGO-UK</td>
</tr>
</tbody>
</table>

## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1a</td>
<td>2014-07-15</td>
<td>ToC and contents skeleton/draft, distributed to few selected partners</td>
</tr>
<tr>
<td>V0.1b</td>
<td>2014-07-28</td>
<td>First draft distributed to all involved partners (WP7)</td>
</tr>
<tr>
<td>V0.2</td>
<td>2014-08-30</td>
<td>First draft version with partners’ contributions</td>
</tr>
<tr>
<td>V0.3</td>
<td>2014-09-16</td>
<td>First draft version distributed for internal review</td>
</tr>
<tr>
<td>V0.4</td>
<td>2014-09-23</td>
<td>Second draft version after internal review</td>
</tr>
<tr>
<td>V0.5</td>
<td>2014-09-23</td>
<td>Ready for PB review</td>
</tr>
<tr>
<td>V0.6</td>
<td>2014-09-29</td>
<td>Approved by PB and ready for delivery</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

1 Executive Summary .......................................................................................................................... 5

2 Introduction ....................................................................................................................................... 6
   2.1 Document Structure & Style .............................................................................................................. 6
   2.2 Transparent Certification and Compliance-aware Process (WP7) ..................................................... 6
   2.3 Methodological guide of the process-specific service infrastructure (Task T7.5) ................................ 8
   2.4 Relation to other WPs and deliverables ............................................................................................ 8

3 Methodological guide .......................................................................................................................... 9
   3.1 Basic Concepts and Terminology ....................................................................................................... 9
      3.1.1 System, Safety and Development ............................................................................................. 9
      3.1.2 Safety Assurance, Assessment, Certification .......................................................................... 10
      3.1.3 Safety Assessment/Certification Management Infrastructure ................................................. 11
      3.1.4 Safety Related Standards and Processes .................................................................................. 12
   3.2 OPENCOSS Concepts and Terminology ......................................................................................... 13
      3.2.1 Assurance Project .................................................................................................................... 13
      3.2.2 Reference Framework ............................................................................................................. 14
      3.2.3 Baseline process model/framework .......................................................................................... 14
      3.2.4 Equivalence Map ..................................................................................................................... 15
      3.2.5 Compliance Map ..................................................................................................................... 15
      3.2.6 Prescriptive Knowledge ........................................................................................................... 16
      3.2.7 Wrap up ................................................................................................................................... 16
   3.3 Usage Scenario .................................................................................................................................. 16
   3.4 The OPENCOSS BPN - Basic Process Notation ............................................................................. 18
      3.4.1 Basic Process Element (Activity) .............................................................................................. 18
      3.4.2 Network of Activities .............................................................................................................. 19
      3.4.3 Hierarchical Decomposition ..................................................................................................... 20
      3.4.4 Verification and Validation Processes ....................................................................................... 20
      3.4.5 Amplifying Processes ............................................................................................................ 21
      3.4.6 Multiple (Array of) Processes .................................................................................................. 22
      3.4.7 Alternative (Conditional) Processes ......................................................................................... 22
      3.4.8 Other Process properties/characteristics .................................................................................. 23
   3.5 Mapping to existing PM formalisms ................................................................................................. 24
      3.5.1 Mapping OPENCOSS to BPMN 2.0 ......................................................................................... 24
      3.5.2 Mapping OPENCOSS to SPEM 2.0 ......................................................................................... 25
      3.5.3 Mapping OPENCOSS to UML/SysML .................................................................................... 25
      3.5.4 Mapping OPENCOSS to PMOD (ESA) .................................................................................. 26

4 Pilot Deployment .................................................................................................................................. 28
   4.1 Pilot modelling ................................................................................................................................. 28
   4.2 Pilot conclusion .............................................................................................................................. 33

5 Conclusions .......................................................................................................................................... 34

6 Abreviations and Definitions .............................................................................................................. 35

7 References ............................................................................................................................................ 36
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>A computer with wheels</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Example of system development platform and OPENCOSS</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Standard layers</td>
<td>13</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Equivalence Mapping</td>
<td>15</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Prescriptive Knowledge</td>
<td>16</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Basic process element</td>
<td>18</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Basic process element</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Example of a process fragment</td>
<td>19</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Example of an integration process</td>
<td>20</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Example of a V&amp;V process element</td>
<td>21</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Example of an amplifying process element</td>
<td>21</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Example of a multiple process element</td>
<td>22</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Example of a conditional process element</td>
<td>23</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Example of BPMN notation</td>
<td>24</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Example of corresponding OPENCOSS BPN notation</td>
<td>24</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Example of SPEM notation</td>
<td>25</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Example of corresponding OPENCOSS BPN notation</td>
<td>25</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Example of UML/SysML notation</td>
<td>26</td>
</tr>
<tr>
<td>Figure 19</td>
<td>PMOD quality gate notation</td>
<td>27</td>
</tr>
<tr>
<td>Figure 20</td>
<td>OPENCOSS BPN quality gate notation</td>
<td>27</td>
</tr>
<tr>
<td>Figure 21</td>
<td>CENELEC 50128 Process</td>
<td>28</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Software life cycle</td>
<td>29</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Software development</td>
<td>29</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Software requirements</td>
<td>30</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Software architecture and design</td>
<td>30</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Components development</td>
<td>31</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Component design</td>
<td>31</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Component testing</td>
<td>32</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Software integration</td>
<td>32</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Software validation</td>
<td>33</td>
</tr>
</tbody>
</table>
1 Executive Summary

The overall objective of WP7 is to define and implement a safety (assessment) certification management infrastructure (set of integrated tools) to support, with some level of automation and “intelligence”, a “compliance-aware” and “transparent” assessment/certification process.

By compliance-aware we intend a tool support based on a formalized process description (process model) as to promote systematic and complete assessment of compliance with respect to the process model.

The infrastructure “knows” the standard and helps the assessor in assuring compliance.

By transparent we intend the visible and recorded sequence of assessment activities in such a way that it is easy for all stakeholders to see what analysis have been performed and to identify what are the results of these analysis.

The infrastructure records and keeps track of the certification process, thus supporting repeatability and audit.

The assessment/certification process often occurs at the end of the life cycle as the final (closing) step of the development process, but it should better parallel the development process from the very beginning in order to provide early warning of problems and deviations, and to minimise the schedule impact in a sort of concurrent certification (along the same metaphor of concurrent engineering).

The assessment/certification process is therefore intended to be tightly “interwoven” with the development process by allowing developers to assess where they are with respect to their duties to conform to safety practices and standards, in terms of the progress of the work and its level of achieved compliance. Certification is therefore not assumed to be a single event in time occurring just-before-delivery, but rather it is supposed to accompany pro-actively the development life-cycle by flagging deviations and encouraging conformance.

A compliance-aware and transparent certification process will also enable to produce specific metrics for safety-assurance (e.g. level of compliance). In doing so, WP7 fulfils (or contribute to fulfil) the OPENCOSS Objectives ST4 (Transparent Certification) and ST5 (Compliance-Aware Development Process).

This deliverable is the outcome of the WP7 - task T7.5 devoted to develop a methodological guide to use the safety certification management infrastructure developed in this same WP7, and will run a small pilot to evaluate the implementation before integration in WP3.

This guide contains a set of rules to use the platform, usage scenarios with detailed steps, and a mapping to existing formalisms for process specification. The pilot consists of an early assessment of the platform that will evaluate the performance and metrics defined in T7.1.
2 Introduction

2.1 Document Structure & Style

The document adopts a presentation style extremely direct and plain, being targeted to a potential wide audience even outside the consortium.

The document provides an overview and confirmation of basic concepts and gently introduces the new concepts identified by OPENCOSS.

Following an Executive Summary given in Chapter 1, Chapter 2 (this same chapter) provides a general introduction also providing the context and purpose of this guideline in the framework of the overall OPENCOSS project.

Chapter 3 provides the methodological guidelines and specifically section 3.4 presents the OPENCOSS Business Process notation. Section 3.5 show a comparison of OPENCOSS vs. other modelling approaches such as BPMN, SPEM or PMOD (ESA).

Chapter 4 reports about a pilot application.

Finally Chapter 5 draws the main conclusion of the work.

2.2 Transparent Certification and Compliance-aware Process (WP7)

This chapter is a quick remind of the work package WP7 purpose and objectives.

WP7 aims at defining and implementing a safety certification management infrastructure to support the certification process. In a sense it is a new breed of tool (an infrastructure) aimed specifically at supporting the safety assessors but also the same providing benefits to development teams.

In fact the certification process is closely interwoven with the development process by allowing developers to assess where they are with respect to their duties to conform to safety practices and standards, and still to motivate them to see the actual progress of the work and level of compliance being achieved.

We can refer to this infrastructure as a sort of “safety assessment assistant”.

The WP7 defines part of the OPENCOSS safety certification management infrastructure together with WP6. While WP6 centres on the evidence chain as collected from different sources (including development tools), WP7 is more concerned with the processes view of certification (i.e. the specification and execution of the certification process).

WP7 has the following specific objectives:

- Analyse and assess the state of the art and state of the practice in terms of approaches for certification process specification and execution, also by looking at the development and assurance processes as well. Evaluate Agile approaches and continuous integration approaches.
- Develop detailed technical requirements by refining the high level requirements defined in WP2. This includes the identification of business models and constraints such as legal and technological.
• Identify metrics for the certification and safety assurance processes with the pursuit of dependability as a balancing of costs and benefits and a prioritization of risks.
• Design and implement a set of OPENCOSS platform services for certification life-cycle support, standards-compliance awareness, traceability management of certification requirements, and event triggering infrastructure for certification compliance.
• Provide a methodological guide to integrate the OPENCOSS platform services into other existing ALM or tool integration platforms.

WP7 aims at fulfilling the OPENCOSS Objectives ST4 (Transparent Certification) and ST5 (Compliance-Aware Development Process).

**ST4: Transparent Certification Process**

The lack of performance metrics and of certification efficiency and effectiveness estimations limits the capability to assess long-term costs, savings and benefits associated with safety-critical system development and subsequent re-certification activities. OPENCOSS aims at tackling this limitation by providing the necessary infrastructure to follow a systematic and transparent certification process. The principle is to make the certification process explicit and interwoven with the development process, although highly independent and unconditioned from it. An explicit certification process will enable to determine and compute specific metrics for safety-assurance and certification processes.

Such an infrastructure also intends to provide stakeholders (including customers and users) with information about the safety assessment process (e.g., effort and schedule for certification tasks) and the assurance artefacts themselves (e.g., claims, arguments and evidence) as a way to improve credibility.

*Indeed, economists have established that if consumers cannot reliably observe quality before they buy, sellers may get little economic benefit from providing higher quality than their competitors, and overall quality can decline.* The willingness of an embedded (sub) system supplier to provide such data, and the clarity and integrity of the data that the supplier provides, will be a strong indication of its attitude to safety assurance and transparency.

Suppliers should address consistently savings achievable from cross-acceptance and re-use of previous certifications, as well as cost and time reductions due to process standardisation and readiness and transparency of the certification outcomes. For instance, a compositional certification can improve re-certification over the total lifespan. On the other hand a monolithic set of data may be cheaper up front, but much more costly in the long term.

**ST5: Compliance-Aware Development Process**

Addressing the development life cycle workflow is one of the objectives of the OPENCOSS project. Cost-efficient system certification demands a continuous compliance-checking process by enhancing integration of certification goals and development life cycle workflow. The goal is to allow developers to assess where they are with respect to their duties to comply with safety practices and standards, and still to motivate them to see the effective progress of the work and the level of compliance being achieved.

OPENCOSS aims at introducing a tool infrastructure to help keeping the certification evidence up-to-date. Such a tool infrastructure allows for faster certification by automating many of the activities required for certification, so every change triggers (in a sort of impact analysis) a complete run of these activities, signalling those that need to be performed manually. From a process workflow one can infer a temporal and causal dependency between processes, activities and artefacts. For example, editing a requirement shall always precede the verification of that requirement, and the production of the document containing
the list of requirements shall always follow the editing and verification of requirements. It is thus possible to infer a set of rules which can be used to check automatically that the workflow has been followed (enacted) and provide evidence of the level of compliance against safety assurance practices. This is also called “process enactment”.

This is one field where Agile approaches can be used and beneficial. The question is on how we integrate agile approaches into the current standard-based approaches used in a critical system development. Agile processes, when applied with rigour and discipline, are not in contrast with the goal of assuring safety. On the contrary a highly iterative process assuring safety at each step (“sprint”, in Agile/SCRUM jargon) may combine the benefits of an incremental approach with the rigour of a safety assessment.

The project will define common processes enabling partial automation of the certification across organisations, taking into account the business constraints of the stakeholders participating in these processes.

2.3 Methodological guide of the process-specific service infrastructure (Task T7.5)

This chapter is a quick remind of Task 7.5 (part of WO7) purpose and goals.

The task T7.5 develops a methodological guide (this same document) to use the safety certification management infrastructure developed in this WP7, and will run a small pilot to evaluate the implementation before integration in WP3. The guide will contain a set of rules to use the platform, usage scenarios with detailed steps, and a mapping to existing formalisms for process specification. The pilot consists of an early assessment of the platform that will evaluate the performance and quality metrics defined in task T7.1.

This task foresees one deliverable (this same document):

D7.6: Process-specific service infrastructure: Methodological Guide: The guide will contain a set of rules to use the WP7 tool infrastructure, usage scenarios with detailed steps, and a mapping to existing formalisms for process specification.

2.4 Relation to other WPs and deliverables

This document relates mainly to the following other OPENC0SS deliverables:

- **D4.4 - Common Certification Language: Conceptual Model** - This document uses the Argumentation meta-model described in D4.4, and models conforming to the CCL meta-model as inputs into the arguments. It provides the foundation for processes and artifacts definitions and their relations.
- **D1.1 - Constraints of the certification process.** - It provides the foundations for the certification process across multiple domains.
- **D2.2 - High-level requirements** – It provides the high level requirements for the OPENC0SS Platform
- **D3.3 - Integrated OPENC0SS platform** – It describes the OPENC0SS Platform and is the basis for this document acting as baseline for it use for process specific aspects.
3 Methodological guide

3.1 Basic Concepts and Terminology

3.1.1 System, Safety and Development

The term **Safety Related System** (many people prefer the old term **Safety Critical System**) identifies a system that “may potentially” cause harm/damage during its operation. We restrict our focus here to E/E/PE (Electric/Electronic/Programmable Equipment) (as does IEC 61508). And we focus only on functional safety, not considering therefore mechanical robustness, explosive material, chemical leakages, etc.

Please note that a modern electric vehicle may almost qualify as an E/E/PE system! A “computer with wheels” is often claimed.

![Figure 1 - A computer with wheels](image)

The term system is here used as the **system of interest** to be developed. For a car maker the system of interest is a full vehicle. For a tier 1 the system of interest may be a Break System. For a tier 2 may be a single ECU (Electronic Control Unit).

**A Safety Related Development Project** is a set of activities, people, tools, etc. aimed at developing a **safety related system**. We cover the whole development life cycle, from conception to retirement (as illustrated by ISO 15288 for system development). Software is part of the system (usually embedded into it) and has its specific software life cycle anchored to, and consistent with, the overall system life cycle (as illustrated by ISO 12207 for software development). See 15288 appendix E for ISO 12207 process alignment with ISO 15288. These two standards are however relatively general and “safety agnostic”. They are therefore complemented/augmented by a safety specific standard as ISO 26262.

**A System Development Platform** provides a set of tools supporting a system development project. Examples of system development platforms are Polarsys/Topcased (open source) or IBM Rational Suite (commercial). Often, a platform is just a collection/basket of heterogeneous tools with some level of integration. Synonymous are System Development Environment, System Engineering Environment or simply Tool Chain. The term “platform” has been preferred here to better focus on tool integration.

The development of a safety-related system includes specific activities and goals aimed at safety achievement that are not applicable for a non-safety related system. We collectively call these activities simply as **Safety Assurance or safety engineering**.
Safety assurance cooperates with other disciplines and usually places additional requirements (called safety requirements) that will be allocated to design elements (called sometimes safety mechanisms), that in turns will be verified and validated (safety V&V). A traditional life cycle becomes therefore a life cycle safety-aware.

In order to support a life cycle (safety-aware), a good system development platform should include additional functionalities and tools which are intended to support exactly safety engineering (e.g. FTA tool). We may then say that we have a System Development Platform (safety aware).

OPENCOSS is not intended to replace or compete with any System Development Platform nor to provide specific safety tools (e.g. FTA, HARA, etc.). OPENCOSS provides something new and complementary to it.

3.1.2 Safety Assurance, Assessment, Certification

Safety Assurance is the application of safety engineering practices, intended to minimize the risks of operational hazards of a system. It is a self-standing discipline not to be confused with Technical Engineering, Quality Assurance, Reliability Assurance, Maintainability Assurance, Security Assurance, etc. Many organizations have departments called RAMS (Reliability, + Availability +, Maintainability + Safety). We may envisage in the new future even RAMSES (RAMS + Environment + Security). Of course there are relationships among all these disciplines but they are definitively not the same as argued by Nancy Leveson: “a car that does not start is perfectly safe”.

ISO 15288 (Annex D) calls all of these disciplines as “specialty” engineering as they address a specific engineering interest and introduce the concept of process viewpoint which is not far away from the OPENOCOSS approach focused on safety.

Safety assurance helps developing a safety related system by identifying additional safety requirements and appropriate design solutions.

According to SAFE SPICE and also to SAFE CMMI Safety Assurance includes at least a safety Management process, a safety engineering process, and a safety qualification process.

The on-going ARTEMIS project SESAMO, coordinated by INTECS, is deeper investigating relationships between Safety Assurance and Security Assurance and is confirming that the two disciplines often proceed hand-in-hand, but sometime they even conflict, and a trade-off is necessary.

Safety Assessment is a comprehensive and systematic investigation and analysis of all aspects of a system to provide a body of evidence and rationale that shows that an item is justified as being safe within allowed limits on risk. It includes evaluation of product characteristics but also the process used to develop it. Process includes in its broadest sense activities, methods, techniques, procedures, people, tools, etc.

Safety assessment “demonstrates” that a safety related system is (or will be) safe.

In order to be effective and unbiased safety assessment should be independent from system development and also from safety assurance.

It is also recognized the importance to have safety assessment to run in parallel to the development to get deeper insight and to provide early warnings.
The confidence on the achieved safety of a system is a combination of:

- assessment that the “product” conforms to given product safety requirements
- assessment that the “process” conforms to given process safety requirements

It is widely recognised that this combined approach is superior to provide confidence on the system safety compared to analyse in isolation only the process or only the product. Modern international safety standards set forth requirements for both process and product.

Safety Certification is the ultimate claim/declaration by a third party that safety assessment has been properly conducted and has achieved the confidence required. Certification is often required, but not always (see ISO 26262). Safety certification usually culminates into a safety certificate that authorizes the use of the system.

Some time in OPENCROSS the term assessment and certification have been used grossly as synonyms. This is because the certification is always based on a strong technical assessment and often has only an official, formal role.

3.1.3 Safety Assessment/Certification Management Infrastructure

The safety assessment (and certification) task is often conducted by high qualified persons (assessors), but with little or no support by any tool. Most of the cases safety assessment is just a systematic examination (manual reading) of documents and other kind of records and evidences. Assessor get equipped with large spread-sheet at least to track what has been checked vs. a requirements list (essentially acting as a checklist).

Some of these spread-sheets become more and more elaborated as to provide summaries, graphs, etc. Some spread-sheets therefore become genuine “tools” to support the assessor, though having all limitations of a general purpose office automation tool. Only a few attempts are reported to develop specific assessment tools. One example is the Appraisal Assistant from Software Quality Institute, Griffith University intended to support assessment against CMMI and SPICE models. The tool has some success but limited to the Software Process Improvement community.

OPENCROSS has developed a fully-fledged infrastructure or simply the OPENCROSS Platform, which is aimed and designed exactly to support the safety assessment, and is intended to be complementary to a System Development Platform.
The **OPENCOSS Platform** is composed of a number of **workbenches** in turn composed of a number of **tools**. In particular:

- **Workbenches** support only one or a few activities. Example: “Evidence Management” workbench.
- **Tool** support only specific tasks in the software tool process. Example: “Evidence Analysis”.

### 3.1.4 Safety Related Standards and Processes

A Safety Related development project, depending on the application domains, always adopts a given published and recognised **domain specific safety standard**. Examples of domain specific safety standards are CENELEC, DO178, ISO 26262, etc. Some are international, some are national, and some are international but having a national interpretation.

A published standard is further tailored and interpreted by each interested development organization leading to an **organization specific safety standards** which, in turn, is tailored to the need of a project leading to a **project specific safety standard**.

Note that within a multi-layered contractual hierarchy, a project specific standard is established (e.g. Eurofighter Standard, Galileo Standard) but each sub-contractor is allowed for some further tailoring, so that we get a contractor-specific safety standard. This acts as the project specific standard for that contractor.

We have therefore the following layers:

1. **domain specific safety standard**
2. **organization specific safety standard**
3. **project specific safety standard**

Only the last one is the one actually adopted and executed/enacted by a project. CMMI calls this last one as the “project defined process”. This process shall prove its conformance to the organization process, which in turn shall prove its conformance to the domain specific standards.
The safety community misses a veritable domain-independent safety standard, though to some extent IEC 61508:2010 has the ambition to play this role more and more in the future. Some initiatives are being launched to establish a common Safety Body of Knowledge (as far as possible domain independent).

![Diagram of standard layers](image)

**Figure 3– Standard layers**

Usually the Quality Plan or Development Plan defines the Project Specific Standard. In safety related system however, it is often within the Safety Plan that the Project Specific Safety Standard is established.

Occasionally, a product has to comply, concurrently at the same time, with two different domain specific standards (e.g. avionics and railway). In this case the project specific standards should result in a combined tailoring conforming to both parent standards (we may call this as multi-standard conformance).

More often a system is developed according to a domain specific standard and at only a later time it is required to move and conform to another domain specific standard. We call this domain migration.

### 3.2 OPENCOSS Concepts and Terminology

#### 3.2.1 Assurance Project

Once an assessor is required to run an assessment on a given safety related development project, she/he “creates a new an assurance project” as the basis for the assessment. An assurance project is intended to contain (or refer) all evidences necessary to run the assessment. Therefore an assurance project has one-to-one relationships with a safety related development project.
As an assessor, I may be working in parallel on many assurance projects. Or many assessors may work concurrently on the same assurance project. Engineers may also access an assurance project to get information and visibility about assessment status and level of compliance.

### 3.2.2 Reference Framework

It is the way OPENCOSS indicates the formalization of a given safety standard in a formal notation (mixed graphical and textual) that we may simply call a **process model**.

It contains a formal description of a safety standard in terms of activities, artefacts, roles involved, techniques, alternatives, safety integrity levels, applicability, etc. And also activity dependencies, artefact traceability, etc. Such a formal description will strongly support assessment of conformance.

**Actually a Reference Framework is a “re-formulation” of a standard by making explicit and visible all its items and properties, minimizing the use of natural language and maximising the use of a formal model as described by the CCL (Common Certification Language).**

**“a structured way of specifying how to comply with a safety standard”**

The purpose of a reference framework is to specify more formally how to comply with a standard.

A reference framework is elaborated by a **safety standard expert**. The OPENCOSS Platform will be equipped with a library of available reference assurance frameworks for the most popular safety standards.

An organization specific reference assurance framework indicates an internal safety standard usually developed by the safety expert of that organization (e.g. **organization safety manager**).

An assurance project (see above) is always attached with at least one Reference Framework. In general, we have a framework for the domain specific safety standard, another framework for the organization specific safety standard and a framework for the project specific safety standards. This last is called “baseline” (see below).

### 3.2.3 Baseline process model/framework

A **baseline** process model or framework is the framework reflecting the project specific safety standard. It has a primary role because it is the one actually executed (enacted) and against which compliance is to be achieved by the development team and assessed by the assessor.

A baseline framework being project specific is a new exercise to be done for each project (though we can reuse those of similar projects) and is usually carried out by the safety assurance team of that project (**project safety manager**).

### 3.2.4 Equivalence Map

Equivalence Mapping indicates important relationships between two frameworks. Item by item the two frameworks are set in relation and a mapping is established. It is a fundamental functionality to establish level of compliance of one framework and another (e.g. organizational and domain specific).
This mapping does not tell that the project is developed in compliance to a safety standard but is fundamental to assess whether the project specific baseline framework is properly compliant with the organization specific one and this to the domain specific one.

An equivalence map is also used for standards at the same level such as two domain specific standards (e.g. DO178C and ISO 26262). The equivalence map provides all details and guidance to move from one standard to another, in both directions.

The following picture provides an example:

![Equivalence Mapping Diagram](image_url)

**Figure 4– Equivalence Mapping**

### 3.2.5 Compliance Map

A compliance map indicates the level of mapping of given available artefacts and the expected ones from the reference framework. It is a fundamental indication of the progress on the safety achievement and the level of compliance reached.
3.2.6 Prescriptive Knowledge

Prescriptive Knowledge indicates the rules “prescribed” by a safety standard that have to be obeyed in terms of activities, artefacts, roles; rules that have to be complied with. All standards have a substantial amount of prescriptive knowledge which is fundamental for compliance assessment. Other kind of knowledge (e.g. declarative knowledge) is also useful but is not directly connected with compliance assessment.

The figure below represents how prescriptive knowledge affects Reference Frameworks and Equivalence Mapping.

![Figure 5 – Prescriptive Knowledge](image)

3.2.7 Wrap up

An assurance project has the following:

- **Assurance Project Management**: manages the baseline reference framework (for the project specific process) connected to other frameworks with respective equivalence mapping
- **Evidence Management**: manages the evidences (artefact) collected (with a compliance map)
- **Process Management**: manages the process information collected (who did what, when, etc.)
- **Argumentation Management (*)**: manages the argumentations developed (e.g. an argumentation model in the OPENCOSS notation)

(*) Not discussed in this guideline, see D5.6 Compositional Certification Framework: Methodological Guide.

3.3 Usage Scenario

We assume that a Safety Related Project has just started and has adopted a System Development Platform (safety aware); it shall conform to a project specific safety standard that is a valid tailoring of an organization safety standard which in turn is a valid tailoring of a domain specific safety standard.
For the purpose of safety assessment (and ultimately for safety certification) the project is paralleled with an OPENCOSS platform. The primary user of this platform is the designated safety assessor, though it may also be accessed by engineers to verify progress and compliance.

**Activate the tool and Kick-Off the assessment!**
The assessor, though fully independent, participates to the project since the beginning and creates an OPENCOSS assurance project associated with the given project.

**Initialise the applicable standards to comply with**
This project needs to be populated at least with:

1. A domain specific reference framework (likely already available as a library within OPENCOSS)
2. An organization specific reference framework (likely already available in OPENCOSS as prepared by the organization safety manager). This framework has an equivalence mapping to 1).
3. A project specific reference framework (called baseline). This has to be elaborated from scratch or as a variant of a previous project. The project safety manager is in charge of it elaboration and to provide an equivalence mapping to 2).

**Assess the standards tailoring**
The assessor shall perform a critical step to verify the equivalence mappings both formally (nothing has been lost) and substantially (the tailoring has not deviated by the original intent). This is a fundamental step as the plain compliance to a project specific standard is vanished if this standard does not comply (directly or indirectly) with the applicable domain specific (published) standard.

**Import the artefacts**
The OPENCOSS platform is able import or reference artefacts stored in the System Development Platform. This should occur incrementally as those artefacts are produced/delivered.

**Browse the evidences and assess systematically artefacts**
The assessor can now navigate and work in its assurance project. The assessor can add to any artefact an artefact asset evaluation where to place the detailed results of the artefact assessment, including weaknesses.

It is also possible to associate events to artefacts, reflecting the artefact evolution along the life cycle.

**Change standards, only if strictly necessary**
In theory a project should not change standards during it execution. However if this occurs, changes to the reference frameworks have to be performed and a revision of the equivalence mapping have to be re-assessed. Moreover, executed activities and existing artefacts should be migrated from the old baseline framework to the new one. We warn against a not negligible change to the standards “on-the-fly” (i.e. during project execution) as it may have a significant impact and you may lose integrity of the full process.

**Assess incrementally**
Assessors and engineers work in parallel but independently, each using ad-hoc tools and interacting at key points. The system is incrementally developed and safety is assessed, since the beginning, at each incremental step. Safety assessment is not an after-the-fact activity but an in-process activity to gain deeper insight of technical decisions and to provide early warnings.

Even a modern time boxed approach as the one proposed by Agile fits the above approach, where assessment is incrementally performed at each iteration (sprint in SCRUM jargon).
The OPENCOSS tool drives the assessor to systematically analyse all artefacts available and keeps track of inspection results (e.g. compliant, not compliant, and partially compliant with these limitations....). Everything is stored and recorded. Summary of compliance are readily available.

Gap-analysis functionality is available.

Artefacts not yet examined or that were found not compliant are highlighted. Nothing can be forgotten accidentally and the whole process is transparent and recorded.

**Perform impact analysis**

In case an artefact is delivered in a new version, the tool “knows” the process dependency and highlights all those dependent artefacts that are proposed to the assessor for impact analysis.

### 3.4 The OPENCOSS BPN - Basic Process Notation

#### 3.4.1 Basic Process Element (Activity)

A process is defined as an “...interconnected set of activities...”. We call a single activity a **basic process element**. It is an atomic process, not further broken down into more elementary processes. A basic process is characterised by input (some time more than one) and output (usually only one) and a role assigned to execute the activity (usually only one). A **role** (e.g. system tester) shall not be confused with the particular instance, called **actor** (e.g. Mr. Smith), assigned to a an activity.

![Figure 6 – Basic process element](image)

Red lines mark “input flow” (e.g. artefacts that are consumed) while green lines rake output which are created/generated.

It is recommended to graphically represent the input/output flow from **left-to-right** or alternatively **from-top-to bottom**. The left-to-right one is preferred for the most relevant input/output and the top-to-bottom one for ancillary input and side-effect output such as a “report” or a set of records.
In the very old SADT notation (Structured Analysis and Design Techniques), input arrows coming from the top were intended to be more “controls”, such as a procedure leaving the left for the primary input.

### 3.4.2 Network of Activities

A complex process is defined by “interconnecting” basic process elements, thus elaborating a complex process map (or process model). The example below represents a fragment of CENELEC 501268, Section 7.6.

![Figure 8 – Example of a process fragment](image)

There are no limits on how complex a process could result. But sound guidelines recommend to limit complexity to a relatively small number and the famous 7 +/- 2 is always a golden rule, the Miller rule). Please refer to [http://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two](http://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two).
Process elements should be connected enforcing the flow from left-to-right (or top-to-bottom). There is an advantage to place activities on the diagonal as to allow the network of input/output flow less complex and not too much overlapping.

### 3.4.3 Hierarchical Decomposition

To cope with real-life processes we need a mechanism to handle hierarchical decomposition and to allow either to define a process and then at a later time define it in term of sub-processes (top-down modelling), or alternatively to collect a series of processes and build a super-process (parent-process) by grouping them into a single entity (bottom-up modeling). The example below may be grouped in one super-process called “integration”. By a zoom-in operation one can see the sub-process “inside” this “parent-process” and their detailed interconnection. This abstraction mechanism is essential to cope with real life processes and should be used extensively as an “abstraction” mechanism.

![Diagram of an integration process](image)

**Figure 9 – Example of an integration process**

### 3.4.4 Verification and Validation Processes

A number of processes have the only purpose to “verify” or to “validate” a given input. Examples are document reviews, code inspections, unit test, etc. Actually these processes acts as “quality gates” that allow artefacts to pass or fail and be therefore rejected back for re-work (fixing).

These processes play a crucial role in system development and, in safety critical projects, they may even overwhelm the other processes in term of effort and schedule.

It is implicitly assumed that, in case problems are found by this activity, some effort is performed to remove the detected problems (according to a specific problem resolution process).

As a guideline we recommend to identify the process as a “V&V” activity and to assume that the output is the correct and successfully “verified/validated” input. A special tag in the output name is recommended to clarify this, such as (inspected), (reviewed), (unit tested), etc.
Be aware that the V&V task is usually restricted to “detect” problems, while problem fixing occurs by activating the input re-working from the steps back where the problem originates from (we need to identify the original process or processes that injected the problem).

This problem resolution strategy shall not be mapped explicitly at any V&V activity, but should be described once-for-all in a problem resolution process (as in ISO 12207).

3.4.5 Amplifying Processes

Some processes take an input artefact and their purpose is to modify this input to achieve some specific objectives. We do not generate a new artefact but simply a new version of that artefact with additional characteristics to satisfy additional requirements.

Therefore the generated output is the same as the input one, but manipulated, modified, improved. This situation occurs for some artefacts that are manipulated by safety specific activities and become “safety enhanced”.
3.4.6 Multiple (Array of) Processes

Some processes do not generate a single artefact but an array of artefacts of the same type, that in turn have to be processed exactly in the same way. Source code modules are a typical example of multiple artefacts of the same type. The number of those artefacts cannot be established a priori as they depend on the process execution itself. Therefore a process model should include a convention to indicate “several” artefact keeping the exact number abstract. The following notation is recommended to identify an array of identical processes each using an element of an array of similar input artefacts.

![Diagram](image)

**Figure 12 – Example of a multiple process element**

3.4.7 Alternative (Conditional) Processes

It is often the case that different processes shall be executed/performed depending on given conditions. The process therefore has not a unique flow but can take variations/alternatives. The recommendation is to set alternative processes in parallel and associate them with a visible and explicit pre-condition intended to activate them.
3.4.8 Other Process properties/characteristics

It was anticipated that a process is “...a set of interconnected activities...” (ISO 12207), however it is recognized that some processes such as Project Management, Quality Assurance, Configuration Management, etc. are a special kind of processes called respectively management and support processes (ISO 12207).

A management process is intended to plan, monitor, control and report about the execution of other processes (including itself). In a sense it has as input the other processes plans, status, etc. It can be claimed that this process is a meta-process that allows proper planning, execution and reporting of the underlying “engineering processes” (i.e. a process to manage other processes). See a deeper discussion on Appendix A.

Similarly, the configuration management process is a pervasive “support” process that handles most of the artefacts of the underlying process, placing them under configuration control, providing accounts, reports and creating baselines. It takes as input most of the artefacts.

Even Quality Assurance is a pervasive support process controlling quality of the underlying process and artefacts. It takes as input most of the artefacts.

It would be not appropriate to diagrammatically represent those processes at the same level of the underlying network of the engineering process. These support processes shall be represented separately on a different and separate level.
3.5 Mapping to existing PM formalisms

3.5.1 Mapping OPENCOSS to BPMN 2.0

In the following there is a fragment of BPMN notation where it is apparent the focus on roles and the “swimming pool lanes” dedicated to the roles involved. This approach is very common in business processes (e.g. requisition workflow) where so many roles are involved and the notation highlights the exact activities assigned to each on a separate lane.

However, in OPENCOSS BPN, like in SPEM and PMOD, the need for such a special focus on roles is not required and the data flow is “king”. The equivalent representation in OPENCOSS BPN is given below and it is apparent its simplicity compared with BPNM.

![Figure 14 – Example of BPMN notation](image1)

![Figure 15 - Example of corresponding OPENCOSS BPN notation](image2)
3.5.2 Mapping OPENCOSS to SPEM 2.0

SPEM was expressly designed to map engineering processes and it is extremely similar to OPENCOSS BPN.

![Figure 16 – Example of SPEM notation](image)

The corresponding OPENCOSS BPN is straightforward, as illustrated below.

![Figure 17 - Example of corresponding OPENCOSS BPN notation](image)

3.5.3 Mapping OPENCOSS to UML/SysML

UML and SysML are general purpose modelling languages for software and systems engineering. While neither is a process modelling language, both UML and SysML can be used to model processes. Process behaviour can be modelled using activity diagrams as shown below. This example shows a similar process to that in section 2.9.1 with the addition of Artefact 2 as the output of Activity 2. As with BPMN, swim lanes on the activity diagram are used to associate roles with roles.
The corresponding OPENCOSS BPN is similar to that in section 2.9.1 but with an additional artefact Y as output from activity B.

### 3.5.4 Mapping OPENCOSS to PMOD (ESA)

The PMOD notation was defined by ESA specifically to represent software processes. The OPENCOSS BPN has many similarities with PMOD but in our opinion it is simpler though as effective as PMOD.

The fundamental difference lies in the representations of quality gates (see section above 3.4.4). PMOD adopted the following explicit and special notation to identify a quality gate.
This same process is represented in OPENCOSS BPN as illustrated below. We think that OPENCOSS as simpler and by no means inferior to the PMOD notation.
4 Pilot Deployment

The guidelines presented in this document have been exercised into a full blown application to a real industrial case, the new CENELEC 50128 Edition 2011 (Software Development Life Cycle). This exercise has been successfully revised by Railway experts.

4.1 Pilot modelling

The following diagrams provide an overall modelling of CENELEC 50128 New Edition 2011 from software development in railway safety critical applications. It is assumed a Safety Integrity Level 4 (SIL4).

Complex activity boxes are identified with dashed-lines and refined in subsequent diagrams, while the arrow sign indicates that the flow is meant to return to the originating activity box.

![CENELEC 50128 Process Diagram](image)

**Figure 21 – CENELEC 50128 Process**
Figure 22 – Software life cycle

Figure 23 – Software development
Software Requirements

- System Level Requirements (allocated to software)
- System Level Architecture (with HW/SW interfaces)

Requirements Specification

Overall Test Specification

Requirements Verification

Software Requirements Verification Report

Overall Test Specification

Note that Software Validation Plan is not used!

Figure 24 – Software requirements

Software Architecture & Design

Architectures

Designer

Interface Specification

Integration Test Specification

Design Verification

Software Quality Assurance Plan

Design

Integration Tests
- SW/HW Integration

Note that Software Interfaces are not an input to verification

Figure 25 – Software architecture and design
Figure 26 – Components development

Figure 27 – Component design
Figure 28 – Component testing

Figure 29 – Software integration
4.2 Pilot conclusion

The overall conclusion we can draw is that:

- The guidelines are easily applicable with limited coaching and supervision to model a complex real industrial standard.
- The process model is complete and closely reflects the standard.
- The process model is clear, self-explaining, easy to understand and modify, it may even serve the purpose for training on the standard.
- The process model provides a direct visibility of artefacts and their dependencies that are not so evident in the standard. Impact analysis and necessary re-work is made apparent.
- By using this rigorous process model notation, some ambiguity of the standards may be exposed.
5 Conclusions

The OPENCOSS infrastructure is providing an unprecedented level of tool support for one of the most critical activity: assessing the safety of a system in a transparent way and by a tool that “knows” the process and is therefore “compliance-aware”.

However, as for any new tool, proper guidelines are necessary to use the tool in the appropriate way and get the maximum benefit out of it.

These guidelines contain basic rules, recommendations, hints based on the experience of the authors. However, only an extensive use of the OPENCOSS Infrastructure will allow to fine tune these guidelines.

This document has to be considered only as a first edition of a more general OPENCOSS Infrastructure Guidelines for Use that should complement and parallel the OPENCOSS Infrastructure Reference Manual.
6 Abreviations and Definitions

<table>
<thead>
<tr>
<th>ARTEMIS</th>
<th>Advanced Research &amp; Technology for EMbedded Intelligence and Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>CCL</td>
<td>Common Certification Language</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business process Management Notation</td>
</tr>
<tr>
<td>PMOD</td>
<td>Process MODelling (defined by ESA)</td>
</tr>
<tr>
<td>SPEM</td>
<td>Software Process Engineering Modelling</td>
</tr>
<tr>
<td>QM</td>
<td>The Qualifying Machine</td>
</tr>
</tbody>
</table>
7 References

[14] DO 178/C and its supplements
APPENDIX A - About artifacts layers

It was discussed above that not all processes operates at the same level. Some processes (management) actually operate at a higher level, while others (Quality Assurance and Configuration Management) operates as support processes at a lower level.

During the York meeting it was discussed that also artefacts are not always at the same level. For instance most artifacts are reviewed by QA, but the QA generated artefacts are not reviewed by the same QA but a higher level QA. The same applies to managerial documents that are approved by higher level senior management. Most artifacts are versioned and their version is justified by change requests or problem reports. But problem reports are usually not themselves versioned, and we do not write a problem report to correct a wrong problem report, we just replace it.

While all artifacts in a project belong to a given set, not all of them have the same nature and properties. We need to layer the artifact as we did for the processes.

To one extreme we have a logical paradox that recall the famous one from Bertrand Russell leading to the improvement of the set theory. Here below its illustration applied to process modeling.

Suppose you allow a document to reference itself, which is perfectly legitimate.

As an example, in a Development Plan you identify the list of deliverables and you get something like:

- ...
- Software Requirements
- Software Design
- Development Plan (this same document)
- Quality Plan
- ...

We now call a document like this as being “self-referential”.

Now suppose we create a project document called NLIST providing the full list of documents which are not self-referential.

Now is NLIST self-referential?

- If it is, than it shall not contain itself and therefore it is not self-referential.
- If it is not, it shall contains itself and therefore it is self-referential.

Hence the paradox, adapted from the influential philosopher Bertrand Russell.