Collaborative Large-scale Integrating Project

OPENCOSS
Open Platform for EvolutioNary Certification Of Safety-critical Systems

Baseline for the process-specific needs of the OPENCOSS platform
D7.1

<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</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Date:</td>
<td>01 June 2012</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>Xabier Larrucea Uriarte</td>
<td>TECNALIA Research &amp; Innovation</td>
</tr>
<tr>
<td>Jérôme Lambourg</td>
<td>AdaCore</td>
</tr>
<tr>
<td>Martijn Klabbers, Luna Yaping Luo, Luc Engelen, Alexander Serebrenik</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>Janusz Studzizba</td>
<td>Parasoft SA</td>
</tr>
<tr>
<td>Alessandra Martelli, Paolo Panaroni</td>
<td>Intecs</td>
</tr>
<tr>
<td>Michel Ilkiewicz</td>
<td>Inspearit (before DNV ITGS)</td>
</tr>
<tr>
<td>Sandra Torchiaro, Fulvio Tagliabò, Alberto Melzi</td>
<td>Centro Ricerche Fiat S.C.p.A.</td>
</tr>
</tbody>
</table>

### Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>2012-03-05</td>
<td>ToC</td>
</tr>
<tr>
<td>V0.2</td>
<td>2012-03-09</td>
<td>Revised ToC</td>
</tr>
<tr>
<td>V0.3</td>
<td>2012-03-13</td>
<td>Revised ToC</td>
</tr>
<tr>
<td>V0.4</td>
<td>2012-05-07</td>
<td>First draft version</td>
</tr>
<tr>
<td>V0.5</td>
<td>2012-05-08</td>
<td>Second draft version</td>
</tr>
<tr>
<td>V0.6</td>
<td>2012-05-14</td>
<td>First full version</td>
</tr>
<tr>
<td>V0.7</td>
<td>2012-05-21</td>
<td>Ready for WP review</td>
</tr>
<tr>
<td>V0.8</td>
<td>2012-05-24</td>
<td>Ready for peer review</td>
</tr>
<tr>
<td>V0.9</td>
<td>2012-05-29</td>
<td>Ready for PB review</td>
</tr>
<tr>
<td>V1.0</td>
<td>2012-05-31</td>
<td>Approved by PB</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

1 Executive Summary................................................................................................................. 8

2 Introduction................................................................................................................................. 9
  2.1 Structure............................................................................................................................... 9
  2.2 Technical context and objectives WP7: Transparent Certification and Compliance-aware Process ......................................................................................................................... 10
  2.3 Technical context and objectives WP7.1: Transparent Certification and Compliance-aware Process Baseline and process-specific requirements .................................................... 11
  2.4 Relation to other WPs and deliverables .................................................................................. 12
     2.4.1 Relation to WP2 High level requirements ...................................................................... 12
     2.4.2 Relation to WP4 Common Certification Language ......................................................... 13
     2.4.3 Relation to WP6 Chain of evidence ................................................................................ 13

3 Transparent Certification process.......................................................................................... 14
  3.1 Introduction on transparent certification .............................................................................. 14
  3.1.1 Critical Systems context .................................................................................................... 14
  3.1.2 System development ......................................................................................................... 14
  3.1.3 Certification ...................................................................................................................... 15
  3.1.4 System Engineering trends for Safety-Critical Systems Developments .......................... 15
  3.1.5 Cost reduction .................................................................................................................. 15
  3.1.6 Benefits ........................................................................................................................... 16
  3.2 Artefacts, Evidences and Metrics ........................................................................................ 16
     3.2.1 The end-product and the work-products ........................................................................ 16
     3.2.2 Documentation versus information items ...................................................................... 17
     3.2.3 Work-products vs. process records ................................................................................ 17
     3.2.4 Work-products as Models ............................................................................................. 17
     3.2.5 Development process vs. Certification Process ............................................................... 18
     3.2.6 Metrics and Safety Metrics .......................................................................................... 18
     3.2.7 Goal Question (Indicator) Metric .................................................................................. 20

4 Compliance-aware development process approaches .......................................................... 22
  4.1 State-of-the-art certification and development process approaches .................................... 22
     4.1.1 Agile process approach ................................................................................................ 22
     4.1.2 SCAMPI ......................................................................................................................... 27
  4.2 State-of-the-industrial-practice certification process approaches ........................................ 32
     4.2.1 Automotive domain: ISO 26262 .................................................................................... 32
     4.2.2 Avionics domain ............................................................................................................. 35
     4.2.3 Railway domain ............................................................................................................. 37
     4.2.4 Cross and other domain ................................................................................................ 39

5 Process description formalisms ............................................................................................. 42
  5.1 SPESM20 .............................................................................................................................. 42
     5.1.1 Description ..................................................................................................................... 42
     5.1.2 Benefits and drawbacks ................................................................................................. 44
     5.1.3 Usability/applicability within the OPENCROSS development and certification approach ................................................................................................................................. 44
     5.1.4 Supporting tool and constraints .................................................................................... 44
  5.2 ISO/IEC24744 ....................................................................................................................... 44
     5.2.1 Description ..................................................................................................................... 44
     5.2.2 Benefits and drawbacks ................................................................................................. 46
5.2.3 Usability/applicability within the OPENCOSS development and certification approach................................................................. 46
5.2.4 Supporting tool and constraints................................................................................................................................................ 46
5.3 BPMN ......................................................................................................................................................................................... 46
5.3.1 Description ............................................................................................................................................................................. 46
5.3.2 Benefits and drawbacks ....................................................................................................................................................... 48
5.3.3 Usability/applicability within the OPENCOSS development and certification approach......................................................... 50
5.3.4 Supporting tool and constraints............................................................................................................................................. 50
5.4 PMOD ........................................................................................................................................................................................ 52
5.4.1 Description ............................................................................................................................................................................. 52
5.4.2 Benefits and drawbacks ....................................................................................................................................................... 54
5.4.3 Usability/applicability within the OPENCOSS development and certification approach......................................................... 55
5.4.4 Supporting tool and constraints............................................................................................................................................. 55
5.5 Petrinets ....................................................................................................................................................................................... 55
5.5.1 Description ............................................................................................................................................................................. 55
5.5.2 Benefits and drawbacks ....................................................................................................................................................... 57
5.5.3 Usability/applicability within the OPENCOSS development and certification approach......................................................... 57
5.5.4 Supporting tool and constraints............................................................................................................................................. 57
5.6 Declarative workflows ............................................................................................................................................................ 57
5.6.1 Description ............................................................................................................................................................................. 57
5.6.2 Benefits and drawbacks ....................................................................................................................................................... 58
5.6.3 Usability/applicability within the OPENCOSS development and certification approach......................................................... 58
5.6.4 Supporting tool and constraints............................................................................................................................................. 59

6 Supporting tools ........................................................................................................................................................................... 60
6.1 Lifecycle supporting tools.......................................................................................................................................................... 60
6.1.1 Qualifying Machine ............................................................................................................................................................ 60
6.1.2 TOPCASED........................................................................................................................................................................... 62
6.1.3 OSEE (Open System Engineering Environment) .................................................................................................................. 63
6.1.4 Concerto ................................................................................................................................................................................ 65
6.1.5 FRASR ................................................................................................................................................................................... 68
6.2 Process description tools............................................................................................................................................................ 69
6.2.1 Eclipse Process Framework .................................................................................................................................................. 69

7 Gaps between the Baseline Survey and the State of the Art ...................................................................................................... 71
7.1.1 Gaps in transparent certification process ............................................................................................................................ 71
7.1.2 Gaps in compliance-aware development process approaches ............................................................................................. 72
7.1.3 Gaps in process description formalisms ............................................................................................................................. 73
7.1.4 Gaps in supporting tools ....................................................................................................................................................... 74

8 Related projects ............................................................................................................................................................................ 75
8.1 FP Projects .................................................................................................................................................................................. 75
8.1.1 FACIT-SME ........................................................................................................................................................................... 75
8.2 ARTEMIS Projects ................................................................................................................................................................. 76
8.2.1 SafeCer .................................................................................................................................................................................. 76
8.2.2 CHESS .................................................................................................................................................................................. 77
8.2.3 RECOMP .............................................................................................................................................................................. 78

9 Conclusions .................................................................................................................................................................................. 81

10 Abreviations and Definitions ............................................................................................................................................... 84
| 11 | References .................................................................................................................. | 85 |
List of Figures

Figure 4-1 Major SCAMPI activities step by step .......................................................................................... 29
Figure 4-2 Overview of the ISO26262 ........................................................................................................... 33
Figure 4-3 Work products extracted from ISO26262 Management of Functional Safety .......................... 35
Figure 4-4 Certification documents in the avionics domain, Conception phase .................................. 36
Figure 5-1: OMG's 4 meta layers .................................................................................................................. 42
Figure 5-2: SPEM2.0 Method Framework .................................................................................................... 43
Figure 5-3: Overview of the main classes of this SPEM2.0 ........................................................................ 43
Figure 5-4: Three areas of expertise defined in ISO/IE24744 .................................................................... 45
Figure 5-5: Overview of the ISO24744 metamodel ..................................................................................... 45
Figure 5-6: BPNM Diagram Elements ........................................................................................................... 47
Figure 5-7. Pool and Lane formalism ............................................................................................................. 48
Figure 5-8. Clear content of Artefacts ............................................................................................................ 49
Figure 5-9. Use of attributes for defining roles and organization ................................................................. 51
Figure 5-10. ADONIS Worksheet example .................................................................................................. 51
Figure 5-11: A net representing the states of a car ....................................................................................... 56
Figure 5-12: tokens and markings ................................................................................................................ 56
Figure 5-13 Behavioural boundary of traditional approach and declarative language ......................... 58
Figure 5-14 The architecture of Declare ..................................................................................................... 59
Figure 6-1 The Qualifying Machine, Overall view ...................................................................................... 60
Figure 6-2 Automatic regeneration of artefacts .......................................................................................... 61
Figure 6-3 Process model handling of the Qualifying Machine .................................................................. 61
Figure 6-4. The OSEE interface .................................................................................................................. 64
Figure 6-5. Concerto and its interaction with external environment ............................................................. 65
Figure 6-6. Specific requirement traceability information .......................................................................... 66
Figure 7-1. The most relevant factors related to transparency certification process ................................ 71
Figure 7-2. The main stakeholders and beneficiaries in the safety certification process ......................... 72
Figure 7-3. Agile approaches are considered to be used ......................................................................... 73
Figure 8-1 FACIT – SME contributions ...................................................................................................... 76
Figure 8-2 SafeCer's main building blocks ................................................................................................. 77
List of Tables

Table 1 ISO 26262............................................................................................................................................ 33
1 Executive Summary

The objective of WP7 is to define a safety certification management infrastructure to support a transparent certification process. This process has to be interwoven with the development and safety assurance processes 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 effective progress of the work and level of compliance. An explicit certification process will enable to produce specific metrics for safety-assurance. In doing so, WP7 will fulfil the OPENCROSS Objectives ST4 (Transparent Certification) and ST5 (Compliance-Aware Development Process).

This deliverable is the outcome of the first WP7 activity devoted to analyse the state-of-art and state of the practice concerning the process-specific formalisms and approaches to specify, develop and certify safety-critical systems.

The analysis has been performed along four different axes:

- The transparent certification process, providing a definition for it, contextualizing it within the development of safety-critical systems, and identifying its evidences in terms of metrics and artefacts;
- The compliance-aware development process approaches, tackling only new approaches as state of the art (the agile approach that is a real change in this area and e.g. CMMI that represents a new trend in quality oriented processes) and looking at the state of the industrial practice in the three OPENCROSS safety-critical industrial domains;
- Process description formalisms;
- Supporting tools, both for life cycle and for process description.

Based on the results of the baseline survey and the review of the state of the art, the gaps between them have been analysed to determine what they are, why and how they can affect the WP7 activities, and whether they can or should be addressed in WP7.

The gaps have been divided into the following categories:
- Transparent certification process
- Compliance-aware development process approaches
- Process description formalisms
- Supporting tools

This document re-affirms the so called issue of “Babel of standards”, partly justified by the different needs of the different domains. Therefore there is a great opportunity the OPENCROSS consortium to bring together those standards in a common framework.

Agile is probably the most visible progress in software engineering of the last decade and it is possible to start talking about an Agile Assessment, though applying Agile methods to safety assessment is not so trivial and should be considered carefully. In order to achieve that, it is fundamental to deploy a certification platform tightly interfacing the development repository, including both development and certification-related artefacts enriched with proper metrics and specific views on those artefacts, and leading to an incremental and evolutionary assessment.
2 Introduction

Safety assurance and certification are amongst the most expensive and time-consuming tasks in the development of safety-critical embedded systems. European innovation and productivity in this market is curtailed by the lack of affordable (re)certification approaches. Major problems arise when changes to or evolutions of a system entail the reconstruction of the entire body of certification arguments and evidence. Further, market trends strongly suggest that many future embedded systems will comprise heterogeneous, dynamic coalitions of systems of systems. As such, they will have to be built and assessed according to numerous standards and regulations. Current certification practices will be prohibitively costly to apply to this kind of embedded system.

The OPENCOSS project aims to devise a common certification framework that spans different vertical markets for railway, avionics and automotive industries, and to establish an open-source safety certification infrastructure. The infrastructure is being realised as a tightly integrated solution, supporting interoperability with existing development and assurance tools. The ultimate goal of the project is to bring about substantial reductions in recurring safety certification costs, and at the same time to increase product safety through the introduction of more systematic certification practices. Both will boost innovation and system upgrades considerably.

This deliverable is the first deliverable of WP7 produced by Task 7.1. Its objectives are to analyse what a transparent certification process is and to provide a survey of compliance-aware development process approaches and of existing formalisms for process specification and execution in order to analyse what can be used in OPENCOSS and how.

2.1 Structure

This document starts in Chapter 2, by recalling the objectives of the OPENCOSS project and of WP7, addressing in particular the OPENCOSS Technical and Scientific (ST) Objectives ST4 (Transparent Certification) and ST5 (Compliance-Aware Development Process), and identifying existing relationships between WP7 and the High level requirements (WP2), the Chain of evidence (WP6) and the Common Certification Language (WP4).

Chapter 3 introduces the need for a transparent certification process within the context of the development of critical systems. Then, an analysis of what would be the extent and scope of the “work product” concept as generated during the execution of a transparent certification process, and how “transparency” can be measured by means of metrics.

Chapter 4 deals with compliance-aware development process approaches, both state-of-the-art and state-of-the-industrial-practice, mostly looking at the software engineering arena. The Agile process approach is described with benefits and drawbacks in this context, and how it can be tailored to fit the certification of safety-critical systems. Within the state of the art category, the SCAMPI [1] method and the ISO 26262 [2] standard are presented. A short presentation of the state-of-the-industrial-practice in the four OPENCOSS domain is presented: automotive, avionics, railway and cross-domain.

Chapter 5 provides general introduction to existing formalisms for process specification and execution: SPEM20 [3], ISO/IEC24744 [4], BPMN [5], PMOD [6], PetriNets [7] and Declarative workflows [8].

Chapter 6 gives a brief overview of supporting tools, both for life cycle (QM - Qualifying Machine [9], TOPCASED [10], OSEE - Open System Engineering Environment [11], Concerto [12] and FRASR - FRamework for Analyzing Software Repositories [14]) and for process description (EPF - Eclipse Process Framework [15]).
In Chapter 7 the gap analysis, based on the review of the state of the art and the state of the practice, is provided divided into the following categories: (1) Transparent certification process, (2) Compliance-aware development process approaches, (3) Process description formalisms and (4) Supporting tools.

In Chapter 8 some relevant research projects are identified and for each one a short description, the relationship with WP7 and the relevance for WP7 is provided.

Chapter 9 draws some conclusions.

2.2 Technical context and objectives WP7: Transparent Certification and Compliance-aware Process

WP7 aims at defining a safety certification management infrastructure to support the certification process. This process will be interwoven with the development and safety assurance processes 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 effective progress of the work and level of compliance. This WP will define 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 recertification activities. OPENCOSS aims at tackling this limitation by providing the necessary infrastructure to follow a 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., times to carry out V&V and 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. They 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 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 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.

OPENCOSS aims at introducing a tool infrastructure to help keeping the certification evidence up-to-date. Such a tool infrastructure will allow for faster certification by automating most of the activities required for certification, so every change triggers 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 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. 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 jargon) may combine the benefits of an incremental approach with the rigour of a safety assessment. It is a challenge that the OPENCOSS project. 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 Technical context and objectives WP7.1: Transparent Certification and Compliance-aware Process Baseline and process-specific requirements

The aim of this task is to analyse the state-of-art and state of the practice concerning the process-specific formalisms and approaches to specify, develop and certify embedded systems. The stress is put in the certification process.

This is one field where agile approaches can be used. The question is on how we integrate agile approaches into the current standard-based approaches used in a critical system development. A possible reference in this area is the Qualifying Machine concept (see forge.open-do/project/qmachine), a testing framework
integrated with process tracking tools for proving qualification evidence to certification entities. OPENCOSS plans to go beyond this project by exploring how we can apply these approaches to carry out certification effectively in an evolutionary process. Another reference in this field is the work in the FACIT-SME project (www.facit-sme.eu).

One of the goals of this project is to facilitate the use of software engineering methods and to provide efficient and affordable certification of these processes according to internationally accepted standards. In addition, this task will develop the high level requirements from WP2 into detailed technical requirements and use cases related process-related services of the safety certificate management infrastructure. A set of performance and quality metrics will be defined for this WP (interoperability, openness, etc.).

This task foresees two deliverables:

**D7.1:** Baseline for the process-specific needs of the OPENCOSS platform: this deliverable;

**D7.2:** Detailed requirements for the process-specific needs of the OPENCOSS platform: The high lever requirements coming from WP2 will be refined in this deliverable. It will contain use cases and functional and non-functional requirements of the platform. Non-functional requirements include quality metrics. It is due at Month 12.

## 2.4 Relation to other WPs and deliverables

### 2.4.1 Relation to WP2 High level requirements

WP2 determines the foundations on which the OPENCOSS platform will be built. This implies that the WP7 will provide an intelligent, automated, and highly customizable safety certification management infrastructure integrated into the development processes and existing development and safety assurance tools. For this purpose, WP2 will investigate the overall environment targeted by OPENCOSS in order to define the first common vision of the scenarios where the OPENCOSS platform will be instantiated and implemented. Subsequently, initial requirements for the whole platform (high-level requirements) and a general architecture will be specified. Finally, WP2 will provide a detailed specification of system use case scenarios by using the OPENCOSS building blocks of its general architecture.

During the specification and implementation of the WP7 infrastructure, the following aspects related to WP2 must be considered:

- **High-level requirements** for the OPENCOSS platform, which must be fulfilled when specifying detailed requirements for the WP7 infrastructure. Consequently, WP7 will refine the high-level requirements for the OPENCOSS platform. Traceability must also exist between the high-level requirements specified in WP2 and the detailed requirements specified in WP7.
- The overall OPENCOSS platform architecture, which will determine the building blocks and whose design must be refined in WP7. That is, WP2 will provide the building blocks of the OPENCOSS platform architecture that have to be implemented in WP7. Their implementation decisions and details for the WP7 infrastructure will be specified in WP7.
- **Usage scenarios** for the OPENCOSS platform, which will indicate what the detailed building blocks of the WP infrastructure, will have to do and how they will have to interact with other building blocks, both from the WP7 infrastructure and the WP4, WP5, and WP6 infrastructures. WP7 infrastructure must allow correct execution of such scenarios.
2.4.2 Relation to WP4 Common Certification Language

WP4 aims at defining a common conceptual and notational framework for specifying certification assets, as a means to discuss abstract notations from different domains. The main goal is to have a common “certification language” (CCL), as a means to get mutual recognition of and to discuss abstract notions from different domains. Using a common conceptual framework for different certification standards facilitates the management of claims, evidences and arguments in a common format, allows patterns of certification assessment to be shared, and supports cost-effective re-certification between different standards.

The concrete objectives of WP4 encompass:
- integrating WP5 outcomes into the CCL to ensure it supports compositional certification,
- establishing a clear methodology to use the CCL in different usage scenarios,
- developing the tools to support the CCL (editors, data import/export, etc.).

WP4 initially identified three conceptual levels for the CCL:
- description of safety cases,
- description of evidence characterization,
- description of compliance management.

The CCL will be used to build domain-specific libraries of certification models, which will act as a knowledge database, providing information about safety-related standards (IEC 61508, EN50126, ISO 26262, ARP4754A, ARP4761, DO-254, DO-178B, DO-297, etc.).

The description of processes in WP7 will be enabled by the CCL framework developed in WP4.

2.4.3 Relation to WP6 Chain of evidence

OPENCOSS W6 and WP7 will be in charge of the definition and development of a safety certification management infrastructure. WP6 will deal primarily with evidence management, whereas WP7 will concentrate mainly on process management. Determining the exact interactions between WP6 and WP7 is an issue that needs to be studied in T2.3 (Design of the OPENCOSS platform architecture). Briefly, WP6 can be viewed as the main work package for dealing with the information needs of OPENCOSS (storage, retrieval, querying). In addition to this, the WP7 infrastructure will have its own needs. For example, WP7 infrastructure will have to manage information about the metrics defined for (provision of) transparency and (checking of) compliance. Nonetheless, it yet has to be determined where such information will be stored within the OPENCOSS infrastructure.
3 Transparent Certification process

3.1 Introduction on transparent certification

Nowadays the lack of transparency in certification processes is a recurrent problem in industrial settings. In fact the transparency concept is related with the poor visibility into the overall architecture of systems, how components are verified and integrated, and finally how the resulting system is certified in accordance to a set of regulations and reference standards. In addition this transparency should take into account the stakeholders’ point of view, and it includes the ability to specify and monitor metrics over the resulting products and production processes as a way for controlling these processes.

Therefore we define “transparent certification process” as the sequence of tasks or activities performed for assessing and verifying a product or a component or a set of processes with respect to a set of criteria in such a way that it is easy for all stakeholders to see what actions are performed and to identify what are the results of these actions.

The existence of an infrastructure to make the certification process explicit and interwoven with the development process is needed. In particular the production of specific metrics for safety-assurance and certification processes should be taken into account. For example all these metrics will help to measure costs related to (re)certification and to reduce uncertainty in these contexts. Other aspects such as requirements traceability and evidence production are also considered in order to reduce the hazard of generating unsafe products.

3.1.1 Critical Systems context

Nowadays Software Engineering is playing a crucial role in Critical Systems development. The dual relationship between hardware and software generates a wide spectrum of issues in the critical system industry. This is the case for safety-critical systems where processes and products are intertwined in order to assure that a system is sufficiently safe in a specific context. One additional dimension of the problem of safety-critical systems is the fact that we can find them in many heterogeneous domains that might each have their own special characteristics. Safety-critical systems development encompasses a wide set of challenges covering methods, tools and frameworks.

Recently several efforts have been invested in new standards related to safety-critical systems developments such as the new DO178C [39], the new CENELEC 50128 [44], the new ISO 26262 and the new IEC 61508. One of the overall goals of OPENCOSS it is “to demonstrate a potential reduction of recurring costs for component/product safety certification across systems by 40% and across vertical markets by 30%”. Therefore we need to identify the recurring activities amongst these standards. However some of these standards are specific for a domain and therefore it makes challenging the reuse of safety assurance and certification artefacts across domains.

3.1.2 System development

It is widely perceived that a critical system development should be based on traditional and heavy-weight methods such as RUP (Rational Unified Process) aiming to establish a set of dedicated processes in order to reduce errors and therefore mitigate risks in the final products. However the use of Agile approaches in these contexts is becoming more usual than what was expected and even some benefits can be extracted from these experiences.
In Europe, Thales has announced the use of object-oriented technologies and agile software development methodologies to optimize their development time. These techniques were applied for DO178B level A for the ADU 3200[75]. In its turn NASA was also involved in the study and application of Agile developments for this kind of critical systems.

### 3.1.3 Certification

Each industry, country and market segment requires that each safety-critical system product is certified with respect to a set of laws, regulations and standards. In fact, certification is associated with product, processes and people. In the context of processes there are several reference standards such as DO 178C, which define a set of good software development practices and system design processes for aviation software. Such software certification approaches prescribe a set of practices with predetermined objectives, as a way to avoid the common pitfalls [76] of software development such as poor requirements elicitation, lack of verification and validation tasks and so forth. On the other hand, product certification approaches are more focused on the features of generated work products. For example, in the context of safety critical systems, IEC 61508 is being used as a basic functional safety standard defining different levels for safety, the so-called Safety Integrity Levels (SIL). But there are other regulations/laws/standards/reference models such as ISO 26262 and so forth. Organisations related with these critical systems are dealing with several standards at the same time. These standards imply the performance of recurring tasks for product/component/process safety certification. In fact this is one of the goals within OPENCOSS project.

### 3.1.4 System Engineering trends for Safety-Critical Systems Developments

Safety-critical systems are usually built based on components in order to generate intermediate or final products. These elements pre-exist or they should be developed in ad-hoc manner. Several Software Engineering techniques can be applied to generate all these components. Model based engineering practices and tools are used to represent systems modules and their functional and non-functional aspects. In this context, success stories, such as Model Based System Engineering at Lockheed Martin MS2 [77], have been developed and published. In addition SE trends are not just applied specifically to critical systems developments. As mentioned before, certification requirements imply one of the biggest efforts in safety-critical systems production, and therefore it should be treated as part of the whole development process from early stages. In this context, components are pre-certified as independent products following a set of specific rules and their verification and validation processes as stated by the appropriate regulations. They are supposed to be tailored to be integrated with other components for generating final products. These final products must be in turn certified with respect to the same or even to other set of standards and regulations. Reuse strategies widely known in the Software Engineering community can be used in order to mitigate the impact of this reuse problem. In this sense, generative approaches used in software engineering discipline are potentially used for dealing with certification in order to provide “transparency” to this process. Model based approaches are also perceived as a way for defining, deploying and managing certified products.

### 3.1.5 Cost reduction

Safety-critical systems are amongst the most expensive industries such as railway, aerospace or automotive. These markets demand that organisations do not only fulfil and deal with an increasing number of regulations/laws/models/standards, but also that they support the evolution over the time of their processes, products and certifications. Organisations aiming to maintain their critical systems are required to invest a huge amount of money not only for maintaining their products including their evaluation and assessments, but also their certifications, their software developments processes and many other aspects of the software engineering. More and more safety-critical systems are being built using
commercial-off-the-shelf (COTS) products. Modern engineering and business practices in the automotive domain use massive subcontracting and COTS-based development that provide little visibility into subsystem designs.

All these situations have associated direct costs not just for certifying the first time but also for recertifying the following times. At this point we also need to take into account the overlapping certifications processes where components are certified several times with respect to different models and the component itself has not modified in any way. Therefore safety-critical system producers are investing more than required. There are other indirect costs related to how to manage their impact in a specific organization and in the current operational processes and products. In addition, there are parts of processes and products that should be evaluated more than once, leading to higher costs. In this context, modular and continuous certification is foreseen as a solution for assessing and certifying parts of the systems and they can be reused in other assessments methods related to other regulations/laws/models/standards. In the assessments context: are we able to reuse assessments results from other models? In the end, the critical systems development industry needs to provide an appropriate return of investment for their business through the application of a broad range of software engineering techniques.

*Understanding the effects of certification and assurance techniques, and choosing the right combination of them to yield systems that meet today’s typically ambitious requirements, has become even more challenging recently given the increasing evolution of safety-critical systems.*

### 3.1.6 Benefits

The identified benefits of the transparency concept defined previously in section 3.1 are the following:

- **System suppliers** are better positioned to build safer systems, incur lower certification costs, and offer their system to the market with a more conclusive proof that the systems are fit for purpose.
- **System integrators** are provided with more transparent means to evaluate the effect of different components on the overall safety of the system and avoid latent integration problems. Such transparency is crucial for properly handling long supply chains of components delivered by different suppliers.
- **Certifiers** are able better implement their rules and the international standards, allowing them to certify systems with a higher degree of trust and at lower costs.
- **End-users** who must optimize the value of their investment are able to get high-quality and more structured information that they can use to select among competing alternatives.

### 3.2 Artefacts, Evidences and Metrics

#### 3.2.1 The end-product and the work-products

A safety development process within this context is expected to produce an “end-product”: i.e. the safe system. However, the development process itself is highly articulated as a set of interconnected activities or sub-processes each producing intermediate “work-products” (also known as “artefacts”). Work-products include requirements specification, design documents, test specifications, etc. Some of these work-products are contractually agreed to be deliverable as accompanying documentation associated with the end-product. Other work-products remain as internal information mainly for the purpose of supporting future maintenance activities.
OPENCOSS shall address both end-products and work-products. Work-products provide early indications on the final work-product associated with the safe system and also make it possible to perform more accurate safety analysis than working directly on the end-products.

### 3.2.2 Documentation versus information items

In the past, work-products were mainly represented as documents. We moved away from paper (formally signed) documents toward fully electronic documentation. This electronic documentation was however just an electronic reproduction of the paper document (using a word processor). Recently work-products have been evolving and becoming a collection of pieces of information which may be textual, graphical, tabular or a mixture. In a general sense we may say that work-products are evolving from a document-centric approach to a model-centric approach which stresses the nature of atomic pieces of information that are articulated and networked with relationships among them (models).

We may talk about atomic pieces of information (fine grain granularity) such as individual requirements aggregated to form a requirements set instead of talking about a requirement “document”. Word processors are replaced by databases, graphical, diagrammatic and tabular tools. We can conclude that today and tomorrow work-products are no longer plain documents but collection of interrelated information items (models).

OPENCOSS shall directly move to the information centred approach, leaving behind the old document oriented approach. The assessor shall examine information, not documents.

### 3.2.3 Work-products vs. process records

Work-products represent the results of activities and are an input to further activities leading eventually to the final “end-product” as its ultimate outcome. However, some information is accumulated just for the purpose of keeping a record of where, when, and how the activities have been executed. This kind of information is called process records. Often process records are maintained as integral part of documents in the form of date, authors, approval workflow, versions, change records, review reports, minutes, etc. As an example, the minutes of a meeting are not a true work-product but a record of what has been discussed, by whom, where and when. Process records are important for making a process traceable and auditable in order to check process conformance to given requirements (e.g. the safety manager shall attend all reviews of a certain kind, or the test has to be done by staff that is independent from the development staff).

OPENCOSS shall make it possible to access to process records. In many circumstances safety requirements refer to specific process properties (how, when, by whom). Since processes produce work-products, it shall be encouraged that every work-product carries its own process records (who developed it, when, with which tools, in how much time, etc.). In a work-product oriented repository, the process view is a kind of “additional” dimension to navigate (the process view). At any time by selecting a work-product all process information related to it shall be visible.

### 3.2.4 Work-products as Models

In most cases, different work-products may be interpreted as models of the system at different layers of abstraction or from different view-points (a.k.a. perspectives or areas of concern). A set of requirements is a model of the system as seen from the perspective of the customer’s expectations. An architecture is a high level view of the system realization. A test plan may also be considered as a view of the system’s behaviour as expected.
In a sense the system development is an ordered sequence of steps (often iterated incrementally) where we move from a higher functional level view of the system down to its physical and lower level of detail. These models are often accompanied by and interwoven with other models reflecting specific aspects and properties such as a computational model, a quality model, a safety model, a security model, a performance model, an energy model, etc. where each model focuses on the specific properties under investigation. In general we talk also about a mechanical model, an electric model, a behavioural model, a hydraulic model, an optical model, etc.

We may say that we have a core “system” model which represents the system in its purest architectural view. This model is the system backbone which is augmented by a set of related models providing additional information on specific view-points (perspectives, areas of concerns).

OPENCOSS shall definitely embrace the model centric approach. Not only product information shall be expressed as a series of models, but safety itself would require a safety model.

### 3.2.5 Development process vs. Certification Process

The certification process operates “on” the end-products (the safe item under investigation) and its development process. This means a systematic analysis of relevant work-products and process records. From the perspective of the certification process these work-products and process records are *evidences*. Not every development work-product is necessarily an evidence; e.g. a record about effort spent on a given activity may be totally irrelevant to a safety analysis.

The certification process may operate at the end as a final stage of the development process, or better, it should accompany incrementally 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 ideas of concurrent engineering).

The certification process itself produces work-products, process records and a final product (e.g. a certificate).

The purpose of certification work-products and certification process records is to track the certification process itself and make it auditable (who was the assessor, when, does he/she has the proper competence?).

A certification process may itself be the subject of an audit, an investigation, to provide additional assurance that it was performed properly (i.e. in according to assessment standards and plans). This is the syndrome of “checking the checkers”, which pervades all life cycle activities in order to improve our confidence in the outcome (i.e. assure). (This audit may occur also in case of an accident to verify, e.g. during a trial, potential weaknesses of the certification process itself and corresponding liabilities!)

OPENCOSS shall target the identification and description of a safety certification process. The process shall run concurrently and be interwoven with the primary development life cycle. At the same time, work-products and process records of the safety process shall be generated and maintained in a sort of parallel and complementary certification repository.

### 3.2.6 Metrics and Safety Metrics

There is no need to stress the importance of quantitative management of processes, but it is worthwhile to recall some famous statements.
• “In God we trust... all others bring data” (Edward Demming)
• “Anytime you are able to associate a number to something, your knowledge about that becomes of another nature ...” (Lord Kelvin).
• “If you cannot measure it, you cannot improve it” (Lord Kelvin).
• “You can’t manage what you can’t measure” (Tom De Marco).

System development life cycles are more and more managed quantitatively (see also the focus of CMMI level 4). As for software, old standards like ISO 2196 already established a set of recommended metrics (later complemented by ISO 14598 and ISO 25000). At system level the prestigious INCOSE published in 2010 e new edition of System Engineering Leading Indicators.

Metrics are used to provide objective (and quantitative) information about the processes (e.g. effort, schedule, variance), end-products and work-products.

Quantitative (predictive) models are often deployed to “predict,” from early indicators of available processes and work-products, later indicators of the end-product or the whole completed process. This applies mainly to cost, schedule and quality predictive models (such as defect density or reliability, measured as MTBF).

Most of the current safety standards refer two different and complementary approaches:

- **Quantitative** safety approaches for random failures (due to wear out) usually expressed with numbers and metrics such as MTTF (Mean Time to Failure) as in ISO 25119 or FIT (Failures in Time) as in ISO 26262.
- **Qualitative** safety approaches for systematic failures (due to design errors) usually expressed by high level of design assurance, in terms of more stringent life cycle processes as to minimize residual design faults.

In general, for software only a qualitative approach is used, while for hardware both approaches are used. There is however, a lack in the quantitative measurement of global properties like safety. Some use something like MTBA (Mean Time Between Accident). With this kind of measurement we may say that traveling by airplane is safer than traveling by car. Others prefer forms such as MTBH (Mean Time Between Hazards) which may be particularly useful when small numbers of accidents occur. Finally counts of near-misses are also pursued. A near-miss is “an unplanned event that did not result in injury, illness, or damage – but had the potential to do so” (Wikipedia). In fact the fundamental problem is that safety is measured in terms of events (accidents) that we do not want to happen and such events will be very rare in a safe system.

Safety is usually expressed in terms of discrete enumerated safety integrity levels (usually 4 or 5, e.g. from A to E, or from 1 to 4). Required MTBF (or, equivalently, FIT) with respect to safety-related random failure rates is often associated with these levels.

More elaborate hardware architectural metrics refer to, for example, a Safe Failure Fraction (SFF) as the percentage of residual safety-related failure rates over all residual failure rates (in IEC 61508). In the ISO 26262 standard for automotive functional safety, two such architectural metrics are the Single Point Fault Metric (roughly equivalent to the Safe Failure Fraction metric of IEC 61508) and the Latent Fault Metric (which measures the robustness of the architecture with respect to latent multiple-point faults).
In addition to the hardware architectural metrics cited above, which are expressed in relative terms (i.e. percentages), there are absolute metrics on required residual error rates, such as the Probability of Dangerous Failure on Demand (PFD) and Probability of Dangerous Failure per Hour (PFD) metrics of IEC 61508, and the Probabilistic Metric for random Hardware Failures of ISO 26262 (roughly equivalent to the PFD metric of IEC 61508).

Particularly in the context of distributed development, there are techniques for apportioning the metrics into local “budgets” for each developing participant, each of whom must adhere to those local targets which are then later combined into the overall metric calculation.

In many circumstances software life cycle metrics (e.g. cyclomatic complexity, max nesting size, MC/DC test coverage, etc.) may be associated to a specific design assurance level and therefore are used to check safety requirements.

Most safety standards provide requirements of a qualitative nature (e.g. the design must be checked by an independent person), but sometimes also by quantitative requirements (e.g. test shall achieve 100% branch coverage, the global MTBF of that component shall be more than 10^8 hours).

The certification process needs to access some metrics to demonstrate that specific quantitative safety requirements have been achieved. We do not see specific metrics generated by the certification process itself. The state of the art does not permit the determination of an exact safety level, but we use only few quantitative safety levels.

OPENCOSS shall identify typical life cycle metrics that are relevant to some safety requirements and shall support, as far as possible, automatic computation, analysis and reporting.

### 3.2.7 Goal Question (Indicator) Metric

The SEI defines an indicator as a representation of measurement data that provides insight into software development processes and/or software process improvement activities. A measure quantifies a characteristic of an item whereas an indicator may use one or more measures. For example, an indicator may be the trend of a measure over time or the ratio of two measures [81].

Indicators can be used to measure “transparency” in a transparent certification process.

Some of the most recognised methods related to metrics and measurements it is the so-called Goal-Question-Metrics (GQM) and its evolution called Goal-Question-Indicator-Metrics ([21], [22]).
GQM methodology is structured as follow:

1. Conceptual level (GOAL): It represents a layer containing all goals for a project, programme or organisation, and it is agnostic from the quality model or reference standard. It takes into account the existing objects or evidences in a particular environment such as:

   - Products such as deliverables, documents, work products (E.g., specifications, designs, programs, test suites).
   - Processes: set of activities related to the developments processes or certification processes; e.g., specifying, designing, testing, interviewing.
   - Resources: Elements used by processes in order to produce their outputs; e.g., personnel, hardware, software, office space.

2. Operational level (QUESTION): A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.

3. Quantitative level (METRIC): A set of data is associated with every question in order to answer it in a quantitative way. The data can be:
   - Objective: If they depend only on the object that is being measured and not on the viewpoint from which they are taken; E.g., number of versions of a document, staff hours spent on a task, size of a program.
   - Subjective: If they depend on both the object that is being measured and the viewpoint from which they are taken; E.g., readability of a text, level of user satisfaction
4 Compliance-aware development process approaches

4.1 State-of-the-art certification and development process approaches

We have restricted this section to new approaches, without going into details about development processes that have not undergone to significant changes for some time. We restricted our dissertation to two new approaches: the agile approach that is a real change in this area and the CMMI that represents a new trend in quality oriented processes.

4.1.1 Agile process approach

4.1.1.1 Description

Agile software development process has become very popular in recent years. The surveys [25] and [27] of user experience show that majority of the projects (however not all of them) improved their quality, cost and deadlines effectiveness after switching from more traditional models to Agile.

Unfortunately some of its principles like limited amount of documentation or short time-boxed release cycles seem to be unacceptable for many projects including safety-critical systems. Initially Agile process was considered to fit only small applications, while recently it has been used in large systems development.

Adopting Agile in such systems requires adapting the practices in consequence. Fortunately, as it is described below, pure Agile process can be tailored to fit safety-critical certification-driven systems.

1. Agile development values and principles

Agile process follows the guidance expressed by "Agile manifesto" [28] that define the 4 core values of Agile approach:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

and 12 principles:
- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Close, daily co-operation between business people and developers
- Face-to-face conversation is the best form of communication
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Focus on simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances
There are several methods in Agile world (Scrum, Extreme Programming (XP), Test Driven Development, Feature Driven Development) - they all follow the values and principles of the above manifesto.

There are several core concepts of Agile process which are in difference to traditional processes (for example Waterfall Model or V-Model) and do not fully fit standards-regulated projects:
- small number of documentation for requirements specification, design, architecture
- requirement definition, design evolves during whole project cycle
- short development and release cycles
- customer close relations during whole project (not only during initial phase)
- small teams with multi skilled individuals, reliance on self-organization, not following the well-established process - anybody can do anything (one’s roles can be switched from development to testing)

II. Safety standards process features

Below have been listed the core features of development process required by typical safety standard (based on DO-178B and Cenelec EN 50128):

- requirements should be completely defined and documented
- system analysis, design and architecture should be documented - including low level design
- development process should be planned
- testing strategy and process should be planned
- testing results should have high code coverage. Test results should be stored and accessible in reports
- design standards, coding standards should be followed. The status should be visible on reports
- configuration/change management system should be used
- traceability between project artefacts should be recorded and be viewable. This means the need of maintaining more documentation.
- V&V should be performed by independent team

4.1.1.2 Benefits and drawbacks

I. Benefits - How pure Agile fits the Safety standards regulated process

The aim of Agile is similar to safety standards: provide high quality software, which meets customer needs. Agile focuses on intrinsic quality which is obtained by ensuring that the entire development process and practices deliver built-in quality. Therefore, in all Agile approaches there is big emphasis put on testing, it is an integral part of a process:
- acceptance tests prepared based on requirements - from the customer’s perspective.
- automated tests prepared once enough code is written (or in Test Driven Development even before there is any code)
- continuous automated integration and builds during the entire development lifecycle.
- continuous automated tests (unit tests and other) run all the time as software is developed.
II. Drawbacks - Problem with pure Agile approach.

In Agile the resulting working program is the most important, but it lowers the role of thorough requirement definitions, design, analysis - which are core in certification process. Moreover, in order to keep traceability information it requires maintaining thorough documentation.

Fortunately there is a solution - Agile can be adjusted to fit safety-critical systems development.

4.1.1.3 Integration/applicability with development approaches and standards

This chapter presents current state-of-practice of using Agile process in development of safety critical software [23], [24], [29], [30], [31] and [32].

I. How to adjust Agile to meet safety critical systems development

Agile does not treat mission-critical software differently - as usually the stress is put on maximum quality of the final product.

Unfortunately there are key standard process requirements missing - as described in "4.1.2 II. Drawbacks - Problem with pure Agile approach."

Agile can be adopted and can be used in a safety critical environment.

What should be adjusted in this process is:

- more documentation is really needed. Minimal documentation rule should be dropped. Requirements definition, design and architecture should be documented.
- traceability between the documented artefacts is needed. This means even more documentation. If possible this should be automated.
- responsibilities should be separated so that the validation testing and development of specific requirement or functionality are done by separate (independent) team members
- (Optional) Evolutionary Requirements rule (requirements are not fully defined at early stage but they evolve in subsequent development iterations)

According to some articles (e.g. [23]) this should be dropped, because:

- Safety Impact Analysis (SIA) is conducted before the design and coding phases, before SIA all the requirements should be finally specified and do not change
- The Agile practice: requirement change - redesign - refactor - In mission-critical systems refactoring is a risky and costly task due to size and complexity of the systems. Moreover keeping the traceability information up-to-date during heavy source code refactoring would be an additionally challenging task.

However, other articles (e.g. [24]) claim that there can be very generally specified requirements and they can be refined during following iterations, and it is even better to do it gradually, because:

- Safety assessments made in small increments can be more focused, delivering better results.
- Risk analysis has a gradually evolving scope, which should improve its quality.

The rest of Agile process features described in 4.1.1 Description (like short iterations, close customer relationship etc.) stay untouched.
II. Agile process fitting safety certification requirements and certification compliance transparency – the process flow

This chapter describes current state-of-practice of Agile development process and how it can be used in safety-critical project - in order to have the product certified. It also discusses how one can achieve that the process is certification/standard compliance aware - so team is able to view the compliance status at each stage of the process.

The below Agile process can be further adjusted depending on specific project conditions.

- **Requirements initial definition.**
  Team and customer team (stakeholders) actively work with each other - both during initial modelling of project scope, and later during the whole cycle. The requirements can evolve and be refined during the incoming iterations.

  *Certification compliance transparency:* the requirements should be defined and stored (preferably in the management system)

- **QA team prepares acceptance tests** based on the requirements definition.

  *Certification compliance transparency:* there should be traceability registered from requirements to test definition

- **The requirements are split into architecture design and sub-requirements.**
  In pure Agile these are often just hand-sketched diagrams, we will rely on the expertise of team members and reviews (e.g. pair programming in XP) - but in safety-critical system architecture design should be specified in details and stored in traceability system.

  The design can be changed, and can evolve in future iterations.

  *Certification compliance transparency:* The design documents should be stored and traceable from requirements.

- **Guided by our architecture models and sub-requirements a development and tests plan** is prepared and maintained up to date through all the development lifecycle.

  *Certification compliance transparency:* Development plan and tests plan should be traceable to requirements.

- **Developers start coding and preparing automated unit tests** - so that there is a solid base for possible changing the requirements in future iterations.

  Automation of as many of the tasks as possible is required for efficient verification and validation and is crucial in effective Agile:
  - Continuous integration - so that we have current status of the final product always at hand, to assess how it behaves and to learn from it.
  - Prepare automated tests once enough code is written. E.g. in XP all code need to have UT
  - Continuous check of coding standards and rules so that we have current view of the code quality

  *Certification compliance transparency:*
1. There should be traceability information:
   - from source code to the requirement which is implemented by this code
   - from test cases (and resulting test executions) to the requirements showing that the requirement has been implemented correctly

2. Code Review.

For example in XP's Pair Programming practice reviews are performed continuously in pairs. In other Agile approaches we can have separate code review activity. There should be traceability from such review to code.

3. Integration and testing is done automatically - so we have always up-to-date status of our product and its compliance. Results are reported to team members so they always have up-to-date status of their standard-compliance level - during the development process - not at the end of it like in traditional processes.

- All the products (requirements, architecture, code, tests, traceability) will have to be validated in a review process.
  
  *Certification compliance transparency:* Once all the above project artefacts have review status, we can see the overall status of our project.

- **Changes are done to requirements, design, sub-requirements** through iterative and incremental improvements - so the code (and all connected services like db schema) is often refactored.
  
  After such redesign, the automation of product integration and testing ensures the minimum cost of proving that the program is still sound and fully tested. In mission-critical systems **safety should be re-assessed** whenever designs is changed. We need traceability information that ensures safety evidence: requirements to test, code links.

  *Certification compliance transparency:* Having the requirements-design-code-tests traceability information is crucial during such refactoring. It would be the best if it was maintained by some system, which is able to invalidate the traceability dependencies when for example parent requirement or design changes.

- At the end of each development iteration we have a partial but working system to show people.

  *Certification compliance transparency:* Thanks to the automation described in the above steps, we had certification-compliance status view while the iteration was in progress. We have it for the iteration product as well.

- The audit. A final system, a product of many iterations, is ready and should be audited to get standard certificate. The auditor will check the safety evidence, including requirement-code, all the tests (testing code and functionality) will be performed on the final version of the program.
Summary

In the “Certification compliance transparency: ” sections above, for each development process step, we have presented what needs to be additionally performed in the process so that the product of the process can be certified and to have certification compliance information at every stage during the development.

Ideally such functionality should be provided by some process tool - maintaining traceability information automatically with little work overhead to team members.

Full traceability should be continuously and automatically maintained, and be visible for developers, stakeholders and finally auditors.

Conclusion:

Although the applicability of Agile methods with the System-level processes used for safety-critical systems is still to be demonstrated, it seems that Agile process can be adjusted and successfully used for the software development. It focuses on early testing, verification automation which is very important for these systems. However more documentation is needed to store information about requirements definitions, design, and traceability between system elements, in particular, requirements, design items, tests and code.

III. Examples of safety-critical systems developed using Agile.

Below, we list the instances we have found of industrial safety-critical applications that use Agile methods.

i) Medtronic - pacemaker/defibrillator.
   - 50 developers and testers
   - 2-week iterations
   - 15-36 months typical project length
   - 4 agile projects run within past 5 years
   - C++ - several hundred thousand lines

ii) M2S Inc. - PEMS - (Patient Evaluation and Management System) is now the largest radiological registry in the world
   - 6 developers,
   - 4 week iterations, but want to shorten
   - Mix of Perl, Java, Tcl, C, SQ

iii) Thales Avionics – ADIRU for the Airbus A350 XWB
   - DO-178B development at DAL Level A
   - 20 developers
   - Ada2005
   - XP/Scrum/Lean methods

4.1.2 SCAMPI

4.1.2.1 Description

“SCAMPI” stands for “Standard CMMI Appraisal Method for Process Improvement”. This method is developed by the Software Engineering Institute (SEI), who is also the author of the Capability Maturity Model Integration (CMMI) models.

Appraisal requirements
The Appraisal Requirements for CMMI (ARC) is a document defining the requirements for appraisal methods intended for use with CMMI models. The ARC may also be useful when defining appraisals with other reference models. The ARC consists of a set of high-level design criteria for developing, defining, and using appraisal methods based on CMMI models.

Three classes of appraisal methods are introduced:

- Class A (focus on “Institutionalization”): benchmark implementation across the organization,
- Class B (focus on “Deployment”): investigate the process deployment to projects,
- Class C (focus on “Approach”): review the process descriptions.

Class C requirements are a subset of Class B requirements. Class B requirements are a subset of Class A requirements. The degree of confidence in the appraisal results, together with the appraisal duration and cost, decrease from Class A to Class B, and from Class B to Class C.

As stated in the ARC, the general appraisal principles include the following:

- start with an appraisal reference model,
- use a formalized appraisal process,
- involve senior management as the appraisal sponsor,
- focus the appraisal on the sponsor’s business objectives,
- observe strict confidentiality and non-attribution of data,
- approach the appraisal collaboratively,
- focus on follow-on activities and decision making based on the appraisal results.

Appraisal requirements address the activities below:

- appraisal method documentation,
- planning and preparing for the appraisal,
- appraisal data collection, consolidation, and validation,
- rating (Class A only),
- reporting results.

Typical usage modes of an appraisal results are:

- internal process improvement,
- supplier selection,
- process monitoring.

The SCAMPI method
The SCAMPI method is designed to provide benchmark-quality ratings relative to the CMMI models and the People CMM model. It is specified in the Method Definition Document (MDD), in conformance with the requirements defined in the ARC.

During a SCAMPI appraisal, the appraisal team verifies and validates the objective evidence provided by the appraised organization to identify strengths and weaknesses relative to the appraisal reference model. Objective evidence consists of artefacts and affirmations used as indicators for implementation and institutionalization of model practices. Before the Conduct Appraisal phase (see below) begins, members of the appraised organization typically collect and organize documented objective evidence, using defined data collection strategies based on the extent of artefacts available within the organization and aligned with the appraisal reference model.

Roles and responsibilities are defined for the stakeholders involved in a SCAMPI appraisal:

- the sponsor,
- the appraisal team leader,
• the appraisal coordinator,
• appraisal team members,
• selected participants.

The SCAMPI A method processes are listed below, phase by phase and graphically it is represented in Figure 4-1:

• phase “Plan and Prepare for Appraisal”:
  o Analyze Requirements
  o Develop Appraisal Plan
  o Select and Prepare Team
  o Obtain and Inventory Initial Objective Evidence
  o Prepare for Appraisal Conduct

• phase “Conduct Appraisal”:
  o Prepare Participants
  o Examine Objective Evidence
  o Document Objective Evidence
  o Verify Objective Evidence
  o Validate Preliminary Findings
  o Generate Appraisal Results

• phase “Report Results”:
  o Deliver Appraisal Results
  o Package and Archive Appraisal Assets

![Diagram of SCAMPI A activities](image)

**Figure 4-1 Major SCAMPI activities step by step**

The following topics are of particular importance:

• appraisal scope:
  o reference model(s) to be used, and selected process areas in these models,
Throughout results.

The SEI records ratings generated by SCAMPI A appraisals only. The SEI operates a certification program for SCAMPI lead appraisers.

The lead appraiser submits the appraisal results to the SEI. After checking their compliance, the SEI publishes them on the Published Appraisal Results Site (PARS), where they remain available for 3 years. This publication is subject to the authorization of the appraisal sponsor, who remains the owner of the appraisal results.

Throughout a SCAMPI appraisal, caution must be exercised to ensure that confidentiality and non-attribution requirements of the appraisal are maintained. This can be implemented by several means:

- room dedicated to the appraisal team,
- interview sessions conducted in separate rooms,
- strict rules for confidentiality of data and non-attribution of information to appraisal participants; these rules are also communicated to the interviewees,
- no observer (except SEI-certified observers),
- destroy at the conclusion of the appraisal the notes taken by the team members.

### 4.1.2.2 Benefits and drawbacks

As stated in the MDD, SCAMPI A appraisals enable sponsors to:

- gain insight into an organization’s capability by identifying the strengths and weaknesses of its current processes relative to appraisal reference model(s),
- prioritize improvement plans,
- focus on improvements (correct weaknesses that generate risks) that are most beneficial to the organization given its current level of organizational maturity or process capabilities,
- derive capability level ratings as well as a maturity level rating,
- identify risks relative to capability/maturity determinations.

The SCAMPI A method focuses on the processes and the actual practices of the appraised organization and compares them with the expectations of the CMMI models. This strength is leveraged by the fact that
CMMI models are a collection of best practices based on 20 years of experience from worldwide companies and are used by thousands of companies all around the world.

Organisations with a higher CMMI maturity level demonstrate higher predictability and productivity. They lower the risk not to deliver on time, on cost and on quality, which results in higher customer satisfaction and return on investment. A better process helps delivering a better product or service. Though this cannot be taken as a guarantee that a given project will succeed, it mitigates the impacts of adverse conditions.

Certification authorities expect the manufacturers to demonstrate that their products reach an acceptable level of safety. Manufacturers with a higher level of maturity can expect a lower level of involvement of the certification authorities during the project, which reduces the associated workload. Furthermore, certification authorities will benefit from a more predictable schedule. Every time a manufacturer does not meet its commitments, the induced rescheduling penalizes the certification authorities.

A notable constraint is that level rating is permitted in SCAMPI A appraisals only, that is, official appraisals led by a certified lead appraiser. In Class B and Class C appraisals, the appraisal team is not allowed to aggregate elementary observations into more global ones. Concretely, the implementation of a practice of the CMMI model may be quoted “High”, “Medium” or “Low”, but this shall not be used to quote the implementation of a CMMI goal, a CMMI process area or a set of CMMI process areas. Thus, no maturity level can be established for the organisation.

4.1.2.3 Integration/applicability with development approaches and standards

The CMMI for Development (CMMI-DEV) model addresses system, hardware and software development. Though it does not specifically address safety, it can be used in organisations pursuing the certification of their products. In this case, the CMMI appraisers should be careful not to overlook safety aspects, which will be handled among the various requirements that the organisation has to take into account.

A model extension of CMMI-DEV, called “+SAFE”, was developed to specifically address safety. It complements the continuous representation of the CMMI for Development model. Its structure is depicted below.

<table>
<thead>
<tr>
<th>CMMI PA Category</th>
<th>Safety Process Area</th>
<th>Specific Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Management</td>
<td>Safety Management</td>
<td>SG1 Develop Safety Plans</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG2 Monitor Safety Incidents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG3 Manage Safety-Related Suppliers</td>
</tr>
<tr>
<td>Engineering</td>
<td>Safety Engineering</td>
<td>SG1 Identify Hazards, Accidents, and Sources of Hazards</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG2 Analyze Hazards and Perform Risk Assessments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG3 Define and Maintain Safety Requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG4 Design for Safety</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG5 Support Safety Acceptance</td>
</tr>
</tbody>
</table>

The +SAFE extension is laid out similarly as the CMMI models, using the same kinds of components. Specific practices are thus associated with each specific goal to describe activities considered important in achieving the goal.

As an example, the specific practices associated with specific goal SG1 “Develop Safety Plans” are:

- SP 1.1 Determine Regulatory Requirements, Legal Requirements, and Standards
- SP 1.2 Establish Safety Criteria
- SP 1.3 Establish a Safety Organization Structure for the Project
- SP 1.4 Establish a Safety Plan
The CMMI for Development model, with or without the +SAFE extension, can be used to address any development lifecycle (waterfall, incremental, evolutionary, spiral, agile, nested, etc.). Similarly, the CMMI for Acquisition model can be used to address outsourcing, while the CMMI for Services model can be used to address service delivery.

4.1.2.4 Supporting tool and constraints
The SEI publishes its documents as “reports” (ARC, SCAMPI A MDD, CMMI models, +SAFE...) or under various formats (communications, articles, FAQs...). Appraisal teams have to build their own toolset to proceed with appraisals.

4.2 State-of-the-industrial-practice certification process approaches

There is a section for each OPENCOS domain and a section for cross-domain and other domains.

4.2.1 Automotive domain: ISO 26262

4.2.1.1 Description
The last version of ISO26262 [2] has been released in 2011 and it is a standard covering all activities during the safety lifecycle of safety related systems in the context of road vehicles. ISO26262 consists of the following parts:

1. Vocabulary
2. Management of functional safety
3. Concept phase
4. Product development at the system level
5. Product development at the hardware level
6. Product development at the software level
7. Production and operation
8. Supporting processes
9. Automotive Safety Integrity Levels (SIL)-oriented and safety oriented analyses
10. Guideline on ISO26262 (just approved)

Basically this standard specifies the following automotive lifecycle: management, development, production, operation, service, decommissioning. All these phases and different parts of this standard define requirements, which represent the minimum consensus on what has to be done to achieve a safe system, at hardware and software levels.
Figure 4-2 Overview of the ISO26262

There are other management and concept phases and 10 supporting processes and ASIL levels specifications. An overview of this standard is described in Figure 4-2. A general V-Model is the main backbone for an ISO26262 based safety product development, containing in its turn two consecutives v-models for hardware and for software levels.

ISO26262 defines clauses and subclauses and they represent the requirements that should be fulfilled. The following Table 1 ISO 26262 provides and detailed overview of the clauses required per phases.

<table>
<thead>
<tr>
<th>ISO26262</th>
<th>Management of functional safety</th>
<th>Concept Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall safety Management</td>
<td></td>
<td>Initiation of the safety lifecycle</td>
</tr>
<tr>
<td>Safety Management during item development</td>
<td></td>
<td>Hazard analysis and risk assessment</td>
</tr>
<tr>
<td>Safety Management after release for production</td>
<td></td>
<td>Functional safety concept</td>
</tr>
<tr>
<td>Level</td>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>System Level</td>
<td>Initiation of product development at the System level</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specification of the technical safety requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>System design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Item integration and testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Safety validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Functional safety assessment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Release for production</td>
<td></td>
</tr>
<tr>
<td>Hardware level</td>
<td>Initiation of product development at the Hardware level</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specification of hardware safety requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware architectural metrics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evaluation of violation of the safety goal due to random HW failures</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware integration and testing</td>
<td></td>
</tr>
<tr>
<td>Software Level</td>
<td>Initiation of product development at the Software Level</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specification of software safety requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software architectural design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software unit testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software integration and testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verification of software safety requirements</td>
<td></td>
</tr>
<tr>
<td>Production and operation</td>
<td>Production, service (maintenance and repair) and decommissioning</td>
<td></td>
</tr>
<tr>
<td>Supporting processes</td>
<td>Interfaces within distributed developments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specification and management of safety requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configuration management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Change management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verification of software safety requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Qualification of software tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Qualification of software components</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Qualification of hardware components</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proven in use argument</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-3 provides an example of the expected work products per ISO 26262 clause.
The process of the safety assessment in the automotive domain, as previously stated, does not lead to a certification result, but to a conformity assessment and is completely managed by ISO 26262 standard. By this point of view the evolutionary chain of the ISO 26262 standard is already self-aware and does not require specific interpretation, because the source of safety evidences/arguments is the standard itself and, therefore, the safety evidences/arguments are strictly linked to the process.

### 4.2.1.2 Benefits and drawbacks

ISO 26262 standard has a self-consistent structure that alone represents the entire conformity assessment process for safety critical systems in the automotive domain. By this point of view the workflow described in the various part of the standard constitutes itself a complete safety assessment process.

Each requirement of the standard is a step for a complete safety case description.

The main drawbacks for it could derive from the novelty of the standard itself, approved only at the end of 2011 year, that still contains some clauses which would be well defined or linked to the others, requiring a development in a more mature stage; but this will happen consequently to its application.

Now the automotive domain looks at this standard to make effective its application.

### 4.2.1.3 Integration/applicability with development approaches and standards

The integration and applicability of ISO26262 is assured by the fact that in the automotive domain this is the reference standard itself and is tailored on the automotive development processes.

### 4.2.1.4 Supporting tool and constraints

No complete and exhaustive supporting tools now have been developed yet, due again to the novelty of the standard. No specific constraints exist for it, other than the constraints established by the standard itself.

Medini analyze [36] is an integrated tool, developed by ikv (a partner in the OPENCOSS project), for many of the ISO 26262 activities: it implements efficiently core activities of the functional safety analysis and integrates them with existing processes.

### 4.2.2 Avionics domain

The development process in the Avionics domain relies on two main sets of documents:

- the first set comprises the ARP4761/ED-135 [38] and the ARP4754A/ED-79A [37] documents and is used at the system level to design an aircraft, and perform the Safety Assessment.
• At the software and hardware level, the main documents are the DO-178C (software) [39], the DO-254A (hardware) [40]. Several other documents can be applied, depending on development and/or verification techniques used.

Those two sets of document interact with each other via the Design Assurance Level (DAL), assigned to software or hardware components according to the impact a failure of this component may have on the safety of the aircraft.

The DAL ranges from A (Catastrophic) to E (No Effect).

![Diagram of Certification documents in the avionics domain, Conception phase](image)

**4.2.2.1 System level development**

The safety standard SAE ARP 4761 "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment" [38] describes techniques for safety engineering of aviation systems, and thus is often used in association with SAE ARP 4754 “Certification Considerations for Highly-Integrated or Complex Aircraft Systems” [37] and/or other safety standards.

The safety life cycle is split up into the following, all integrated with the wider engineering life cycle.

- Functional Hazard Assessment (addressing hazard identification and preliminary risk analysis)
- Preliminary System Safety Assessment (analyzing the contribution and interaction of the subsystems to system hazards)
- System Safety Assessment (assessing the results of design and implementation, ensuring that all safety requirements are met)

Techniques used in one or more of the above phases include Fault Tree Analysis (FTA), Dependency Diagrams (DD), Markov Analysis (MA), Failure Modes and Effects Analysis (FMEA), Failure Modes and Effects Summary (FMES) and Common Cause Analysis (CCA) (consisting of Zonal Safety Analysis (ZSA), Particular Risks Analysis (PRA) and Common Mode Analysis (CMA)).

The safety standard SAE ARP 4754 "Certification Considerations for Highly-Integrated or Complex Aircraft Systems" deals with the system development processes of aviation systems and how to show compliance to a regulator.

For many safety engineering techniques it references out to SAE ARP 4761 "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment".

### 4.2.2.2 Component (hw&sw) development

The DO-178B/C (EUROCAE ED-12B/C) deals with software for aviation, while the DO-254A deals with the hardware part. The DO standards have a strong focus on the processes used to develop and assess the component being created. Although they are referred to as safety standards, they almost exclusively deal with the development part, requiring different objectives to meet, verification and validation techniques to use depending on the DAL level that has been assigned to the component at the system level. In particular many processes are detailed that need to be performed (the below list is taken from the DO-178C document, applied to software development):

- Software planning process
- Software development process
- Software verification process
- Software configuration management process
- Software quality assurance process
- Certification liaison process
- Certification process

The standard does not prescribe any specific mean to perform those processes. It does not prescribe either the way those processes interact with each other. Instead it lists objectives that those processes need to fulfil, allowing some level of customization in the actual process.

### 4.2.3 Railway domain

The European railway safety standards (CENELEC 5012x series [43], [44], [45] and [46]) have been around for more than 10 years and gained quickly international reputation, as to be adopted as IEC standards giving them world-wide applicability. Due to the high social impact of railway accidents they are among the most prominent and applied safety standards. These standards largely reflects (are a derivative of) IEC 61508 and can be categorised as heavily process based standards.

Differently from the Automotive domain, and similarly to the Avionic domain, railway infrastructures are managed by public national authorities and also an European level authority (ERA European Railway Authority). This encouraged a very rigorous process of independent safety assessment leading to an "authorization to run". Accredited organisations act as "notified body" and are entitled to run independent "safety assessment". The used term is ISA (Independent Safety Assessment).

For instance in France, we have the EPSF Etablissement Public de Sécurité Ferroviaire and in Italy the ANSF Agenzia Nazionale Sicurezza Ferroviaria. Notified bodies include CERTIFER (F), ItalCERTIFER (Italy), Bureau Veritas, RINA, etc.

In general, national regulation and law have a strong impact on the system and its required safety. Up to now, they differ much in different European countries. This makes certification and approval on a cross-country basis more difficult and expensive facing additional requirements. But with the new European view
for opening the market, new practices are put in place: standards that define what is a railway system, what is a subway system, what is an interoperable system, etc.

4.2.3.1 System level development

The main document is CENELEC EN 50126 (first published in 1999). The standard establishes a method for the specification and demonstration of reliability, availability, maintainability and safety (RAMS), for railway domain.

- CENELEC EN 50129 provides general guidance to demonstrate the safety of electronic systems and to construct the safety case for signalling railway application.

- CENELEC EN 50128 provides requirements for the software used in signalling railway application.

- For safety related communication, CENELEC EN 50159-1 is dedicated to closed transmission systems and CENELEC EN 50159-2 is dedicated to open transmission systems.


4.2.3.2 Component (hw&sw) development

Software

The main standard for software development is EN 50128 (first published in 2001). It is a traditional V life cycle process tailored according to SIL in the level of rigour for some technique to apply. Very interestingly it addresses “valid combinations of techniques”, by taking into accounts that techniques interfere one another and they achieve the best effectiveness when used in combination with other techniques. Recently in 2011 the EN 50128 has been revised into a new version after 10 years. It has been endorsed by national authorities before end of April 2012. The Safety Critical System Club published in its latest Newsletter (May 2012 Edition) a good overview of what-is-new in this standard. Here is a brief summary.

SIL 1 and SIL 2 are perfectly identical as well and SIL 3 and SIL 4. No attempt to differentiate. Therefore as a software developer you have only two categories (low and high criticality).

Organization structures are now indicated and marked as “preferred”. Requirements for personnel competence are indicated in greater detail and appendix B identifies key software roles and main responsibilities and associated required competences. For SIL 3 and 4 the option of letting Verifier and Validator to be the same person has been removed. For all SILs the Designer can never be the same person as the Tester or the Integrator.

Section 5.1.2.5 states “The assessor shall be independent of the supplier, or at the discretion of the Safety Authorities, be part of the supplier’s organization or of the customer’s organization.”

The term verification is used in the sense of checking documentation for completeness and correctness, as a kind of quality assurance.

Development tools are now classified, according to a schema similar to ISO 26262 and to DO178C, i.e. tools that does not contribute to the quality of the code (T1, e.g. a word processor), tools used for verification and testing and therefore could fail to reveal defects (T2, e.g. a static analyser), and tools that could create wrong code (T3, e.g. compilers, linkers, other translators). It is interesting to note that validated (or proven-in-use) compilers have been removed from table A4.
Hardware
There is not a specific standard for hardware which is covered by EN 50126. Programmable devices (FPGA) are borderline technologies and often follow EN 50128 as software. Actually you program using high level languages such as VHDL or SystemC.

4.2.4 Cross and other domain
Other application domains are very sensitive to safety and have elaborated their own standards.

4.2.4.1 IEC 61508 Multi-domain (but mainly process automation)
First let’s start with the IEC 61508 [47] which is considered the generic “multi-domain” standard. It has been disseminating the safety culture of electric/electronics equipment world-wide. The standard uses the term PSE (programmable electronic systems) and also E/E/PE (Electric, Electronic, Programmable Equipment). Despite its original goal of becoming a multi-domain standard, it remains rather closely tied to the process automation industry community and culture. This justified the emergence of domain specific standards, more or less derivative of the 61508.

It is worthwhile to mention that this standard has been recently revised with its 2nd Edition in April 2010.

We do not address CENELEC EN 5012x series and ISO 26262 addressed in §4.2.3 and §4.1.3 respectively and we focus on other domains.

The process domain has derived a specific standard: IEC 61511 “Functional safety – Safety instrumented systems for the process industry sector [48].

It is composed of 3 parts:

• Part 1: Framework, definitions, system, hardware and software requirements”
• Part 2: Guidelines in the application of IEC61511–1
• Part 3: Guidance for the determination of the required safety integrity levels”.

4.2.4.2 Nuclear Domain
The Nuclear domain is dominated by few prominent standards:

• IEC 60880 “Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions”, [49]
• IEC 61226 “Nuclear power plants – Instrumentation and control important to safety – Classification of instrumentation and control functions”, [50]
• IEC 61513 “Nuclear power plants – Instrumentation and control for systems important to safety – General requirements for systems”, [51]

All countries using nuclear energy have set up a Safety Authority to control its use and protect the people and the environment from the harmful effects of radiations. In France the Safety Authority is “Autorité de Sûreté Nucléaire” (ASN). Its technical support is “Institut de Radioprotection et de Sûreté Nucléaire” (IRSN). The nuclear power plants are not submitted to “certification”, but to authorization for creation and operation. This authorization is formally delivered by a decree of the government following a proposal of ASN.
4.2.4.3 Space Domain

The European Space projects are regulated by ECSS European Cooperation on Space Standardization. The European Space Agency has undergone a collective effort with national agencies such as CNES, ASI, DLR, etc. to achieve these ECSS common standards. Major projects (e.g. Galileo positioning system) define their own tailoring and interpretation of ECSS (e.g. Galileo Software Development Standards is a major technical achievement). ECSS distinguishes between Quality (Product Assurance), Dependability and Safety. All documents are freely available and downloadable from the internet.

Safety is disciplined by ECSS Q 40 [52]and encompasses:

- human death or injury;
- loss of, or damage to, hardware, software or facilities which could then affect the accomplishment of the mission;
- loss of, or damage to, public or private property; or
- detrimental effects on the environment

Please note that it is not restricted to human life.

Dependability is intended to cover Reliability, Availability and Maintainability (ECSS Q 30 [54]). Essentially it is RAMS without the final S of Safety which is covered in ECSS Q 40. Dependability has to cope with mission loss, major mission degradation, minor mission degradation, etc. with a focus on Fault Tolerance. As for software dependability it is said “…as it is not possible to quantitatively assess these (software) functions, only a qualitative assessment can be made as the dependability of software is influenced by the software development process which is affected by the criticality of the software in relation to mission success”.

All other non-functional properties are covered under the umbrella of Product Assurance (ECSS Q 20 [55]).

Software is fully covered by two companion documents:

- ECSS E 40 Software Engineering
- ECSS Q 80 Software Product Assurance [53]

Other disciplines such as project management and configuration management are covered in the documents of the ECSS M series.

A number of handbooks have been produced as supplement of the standards. Worthily to mention the recent Software Engineering handbook (it includes a nice effort to allows agile approaches in space application and consistent with the ECSS).

4.2.4.4 Medical Devices Domain

This application domain is hampered by a “jungle” of applicable process standards.

- FDA Food and Drug Administration Standards
- IEC 62304 (software life cycle + SILs e tailoring)
- ISO 13485 (Quality Systems Requirements, a kind of tailoring ISO 9001) (replaces ISO 46001) (considered fundamentally equivalent to FDA QSR Quality System Requirements)
- ISO 14969 (it provides a guidance to ISO 13485)
- GAMP 5
- GMP Good Manufacturing Practices
- CDRH Center for Devices and Radiological Health (many guidelines)
- ISO 14971 Risk management for medical devices
• EN 60601 1-4 Programmable electrical medical systems
• EN 60601 1-6 basic safety and essential performance - usability processes for medical device software
• TIR 32:2004 Medical Device Software Risk Management (=> IEC 80002-1)
• IEC 60812 FMEA
• European Council MDD (Medical Device Directive) 93/42/ECC Concerning Medical Devices

Not to mention the many “product” standards applicable to specific equipment.

Of special interest the recent ISO 62304 (published in 2006) for Medical Devices Software Life Cycle. Largely inspired by ISO 12207, introduce 3 criticality levels (A, B and C. the highest) and tailor the requirements according to the level. It supports and an iterative incremental life cycle (Appendix B1.1). It introduces the concept of SOUP (Software of Uncertain Provenance) in order to tackle with pre-developed software with unknown development history.

A special mention to the initiative MediSPICE that intends to harmonize the many standards into a common conceptual framework based on SPICE (ISO 15504) but adapted to the medical devices domain.

4.2.4.5 Agricultural Machines and Tractors Domain
ISO 25119 [69] is the most important standards. ISO 25119 provides safety requirements and guidance on the principles for the design of high risk functional parts of control systems used in agricultural and forestry machinery to ensure human safety. It applies to high risk functional parts of electrical/ electronic/ programmable electronic systems and as part of mechatronic systems. It is a derivative of IEC 61508 and to a large extent it is similar to ISO 26262.

Another relevant standard is ISO 4254 [70].

4.2.4.6 Motorcycles Domain
The community is looking with interest the adoption of ISO 26262 or a variant of it. Large manufacturers such as Piaggio are working in that direction.

4.2.4.7 Industrial Vehicles/Trucks Domain
The community is looking with interest at the adoption of ISO 26262 or a variant of it. Large manufacturers such as IVECo of FIAT Industrial are working in that direction.

4.2.4.8 Marine Domain
The Marine domain is facing the influence of electronics on board (more than 5 MLOK in a 32-meters yacht (according to the Software Engineering Cluster of Excellence). They pursue an initiative called MARRSA (www.marssa.org) to follow an approach very similar to AUTOSAR applied to boats. We are not aware of any specific software development standards or safety standards related to software or electronics.
5 Process description formalisms

5.1 SPEM20

5.1.1 Description

Software & Systems Process Engineering Meta-Model Specification is commonly known as Software Process Engineering Metamodel (SPEM2.0) [3]. It is a UML profile and metamodel for describing processes and methodologies based on UML principles and rigorous metamodeling approach. These principles are summarised in the following figure and it is based on 4 meta layers. From the top layer to the bottom layer there is an “instance of” relationship. MOF (Meta Object Facility) represents the meta-metamodel (M3) and it is used for defining metamodels such as UML (Unified Modelling Language) (M2). UML in its turn is used and instantiated generating class diagrams, activity diagrams, and so on, for describing a system (M1). And finally we generate objects as instances of classes (M0).

![Diagram of OMG's 4 meta layers]

Figure 5-1: OMG’s 4 meta layers

SPEM2.0 defines the following method framework described in the following figure. Basically it defines Method Content and Processes. Method Content is used for defining the building block for a methodology or process such as Roles, Tasks, Work Products and Guidance. These elements are used for building Processes as process components or as delivery processes.
Figure 5-2: SPEM2.0 Method Framework

SPEM2.0 metamodel is based on a wide set of metaclasses but the main elements are represented in the following metamodel snapshot.

Figure 5-3: Overview of the main classes of this SPEM2.0
5.1.2 Benefits and drawbacks

The main benefit of SPEM2.0 is twofold. First we are able to define completely any kind of process or method, and second it can be implemented using most of the UML tools, and even there is an open source tool called Eclipse Process Framework that it is described in a further section. In the context of OPENCOSS, since we partly rely on a model-centric approach, having SPEM2 as a UML profile allows for a seamless integration.

The main drawback is that the industry is not using this metamodel because they are not aware of its existence. From a conceptual point of view the execution of these SPEM2.0 based processes inherits some of the problems of UML2.0 discussed in literature [78], [79] and [80].

5.1.3 Usability/applicability within the OPENCOSS development and certification approach

We are going to use this metamodel as the standard for describing all kinds of processes and methods.

5.1.4 Supporting tool and constraints

In fact we can use Eclipse Process Framework (www.eclipse.org/epf) which is a free tool implementing this metamodel under the Eclipse Public License (EPL).

5.2 ISO/IEC24744

5.2.1 Description

The ISO/IEC24744 [4] is a standard metamodel for development methodologies. It is a formalism for defining and enacting processes. In fact this is “solved by conceiving a metamodel as a model of both the methodology and the endeavour domains” These domains are shown in the following figure. This standard identifies three modelling layers called “domains”, and their relationships are seen as instance-of relationships, in which elements in one layer are instances of some elements in the layer or domain below.
Figure 5-4: Three areas of expertise defined in ISO/IE24744

Figure 5-5: Overview of the ISO24744 metamodel
5.2.2 Benefits and drawbacks

Benefits of this approach are related to the following set of characteristics:

- The communication between stakeholders in the development process (i.e. developers);
- The assembly of pre-existing repositories of method fragments;
- Extension mechanisms for creating metamodel extensions;
- The comparison and integration of methodologies and their associated metamodels; and
- The interoperability of modelling and methodology support tools.

Drawbacks are associated to its usage in industrial scenarios and its scarce supporting tools. In fact commercial tools are not using this metamodel approach and therefore it is seen as a barrier for its adoption.

In addition ISO24744 concepts are not following the strict metamodeling approach used by the OMG (Object Management Group) specifications, so it is hard to be implemented using UML tools.

5.2.3 Usability/applicability within the OPENCOSS development and certification approach

We will investigate how we can use this metamodel within the Opencoss project for describing certification processes taking the advantage of clobjects and powertypes defined within the standard. It is also interesting to check how we can extend this metamodel for this domain.

5.2.4 Supporting tool and constraints

As stated previously there is a limited set of tools within Universities using this metamodel. In fact the existing tools are basically based on UML principles.

5.3 BPMN

5.3.1 Description

BPMN stands for Business Process Model and Notation [5]. It is a standard language for business process modelling that it provides a graphical notation for specifying business processes in a Business Process Diagram (BPD), based on a flowcharting technique.

BPMN derives from an agreement between multiple modelling tool vendors, who had their own notations, to use a single notation for the benefit of end-user understanding and training (Business Process Management Initiative – BPMI). Currently it is maintained by the Object Management Group since the two organizations merged in 2005. The current version of BPMN is 2.0.

The main elements of BPMN notation help the user to build up the complete workflow of a business process, distinguishing the Flow Objects that represent the elements of the workflow, their related connections represented by the Connectivity Objects and the collection of the participants to the workflow who are represented within the Pools and Lanes. Additional Artefacts allow completing the description of
the process. All these elements are described in the following paragraphs. Examples of the diagram elements are shown in Figure 5-6:

![BPNM Diagram Elements]

**Flow objects**
Flow objects represent and describe the core of the workflow.

An Activity represents work that is performed within a business process and can be atomic or non-atomic (compound). The types of Activities that are a part of a Process Model are sub-processes and tasks.

- A Sub-Process is a compound activity that is included within a Process. It is compound in the sense that it can be broken down into a finer level of detail through a set of sub-activities.

- A Task is an atomic activity that is included within a Process. A task is used when the work in the Process is not broken down to a finer level.

An Event is something that “happens” during the course of a business process. These elements affect the flow of the Process: they can start, interrupt or end the workflow.

Gateways are modelling elements that are used to control how the sequence flows of the Activities interact as they converge and diverge within a Process.

**Connectivity Objects**
Connectivity Objects are the connectors between the main elements of the workflow. A process model can contain the following three types of connectors.

- **Sequence Flow** is used to show the order that the activities will be performed in a Process.
- **Message Flow** is used to show the flow of messages between two entities that are prepared to send and receive them.
- **Association** is used to associate data, information and Artefacts with flow objects.

**Swimlanes**

Swimlanes represent the actors (Participants) involved in the steps of the workflow. There are two types of Swimlanes (see Figure 5-7):

- A **Pool** represents the Participants in an interactive Process Diagram. It may be a “black box” or may contain a Process Interaction between other elements of the same type. This interaction, if any, is handled through a Message Flow; Sequence Flow cannot cross the boundary of a Pool (i.e., a Process is fully contained within a Pool).

- A **Lane** represents sub-partitions for the objects within a Pool and often represents organization roles (e.g., Manager, Associate), but can represent any desired Process characteristic. Sequence Flow can cross Lane boundaries.

![Figure 5-7. Pool and Lane formalism](image)

**5.3.2 Benefits and drawbacks**

BPMN focuses on the flow of the activities and on the relationships between individual Artefacts and their association with the activities, allowing a fine (graphical) workflow representation. BPMN is oriented to business/organizational process specification, thus the conformity/certification process can well be managed with it.
Some experiences in the automotive domain by CRF suggest that to model the process with BPMN (e.g. with ADONIS tool by CRF experience) is less time consuming than using SPEM (and EPF tool). The Artefacts produced with BPMN (e.g. with ADONIS tool) provide the capability to show information (e.g. by textual description) beyond the basic flow-chart structure of the Process, as shown in the following example (Figure 5-8).

![Figure 5-8. Clear content of Artefacts](image)

BPMN provides different ways to support information about the resources in the process workflow:

- The association of a particular action or set of actions with a specific resource is illustrated through the use of the Pools and Lanes constructs.

- BPMN offers specific attributes at the Activity level, in which the content of the Activities (Tasks, Sub-tasks), Events, entire Pools and Lanes can be described.

- For user tasks an attribute can be optionally set to identify the person(s), group(s) or organisational units which will perform the task (see Figure 5-9).

Another advantage of process modelling is that it allows for the application of process mining. (Wil M. P. van der Aalst et. al., Process Mining Manifesto, Business Process Management Workshops, 2011, 169-194) Process mining can be used to construct models of processes from event logs, check whether processes as performed in reality conform to their specification, and enhance processes using information retrieved from event logs that result from their execution.

Unfortunately, a standardized formal semantics for the complete BPMN does not (yet) exist. There are, however, a number of attempts at defining such a formal semantics described in literature. (Wong, Peter and Gibbons, Jeremy, A Process Semantics for BPMN, Formal Methods and Software Engineering,355-374 (5256),2008) and (Remco M. Dijkman and Pieter Van Gorp, BPMN 2.0 Execution Semantics Formalized as Graph Rewrite Rules, BPMN, 2010, 16-30) Without a formal definition of the semantics of BPMN, there is room for ambiguity, which is a clear drawback.
5.3.3 Usability/applicability within the OPENCOSS development and certification approach

The language is well usable in the context of the conformity/certification of complex and deeply articulated processes, as ISO 26262 workflow in the automotive domain, because its plain and clear workflow representation. Then activities, work products of reference standards, event, dependencies, roles, organization teams can be clearly described and addressed.

Moreover, the possibility of using the Pools and Lines formalism and the attributes at activity level for describing resources and organizations is quite suitable in a context of certification management, identifying the actors involved for each step of the process workflow. The clear and plain representation of the workflow allows an agile approach to the process steps (e.g. ISO 26262), considering also the recertification: in the automotive field the “recertification” is mainly the use of systems on the shelf, for which the plain workflow represented by BPMN allows an easy “cut and paste” (e.g. process tailoring) of the safety evidences and arguments.

By means of process mining, it can be checked whether a process as performed in reality conforms to its specification in the form of a process model. This means that process mining could be used to show that business processes are executed as they are specified in a standard.

5.3.4 Supporting tool and constraints

A valid (as in CRF experience) supporting tool is ADONIS. The Community Edition of ADONIS (free edition of ADONIS) is well known as professional Business Process Management tool by BOC Group.

ADONIS allows defining an organizational structure, by specifying roles and performers as in the following picture (Figure 5-9).

Yan. Reijders and Dijkman have evaluated five popular BPMN modelling tools (Zhiqiang Yan, Hajo A. Reijers and Remco M. Dijkman, An Evaluation of BPMN Modelling Tools, BPMN, 2010, 21-128). They compared two academic tools, Oryx and DiaGen, and three industrial tools, Biz-Agi, TibCo, and Intalio. They noticed that the difference in functionality offered by the tools does not depend on the fact that a given tool is industrial or academic. Industrial tools, however, are often part of a large BPM suite offering more than just modelling, whereas the academic tools focus on modelling business processes.
Organisational units
You can use Organisational units to describe hierarchical structure.

Roles:
Role represents a particular task range. A Performer may have one or more roles. Many performers may share the same role.

References:
All references to a certain resource can be easily identified. This allows easy analysis of inter-relations in the organisation.

Performers:
Performer is a real person that belongs to at least one organisational unit. You can also state (with an additional arrow) that a certain performer is also the manager of a unit.

Figure 5-9. Use of attributes for defining roles and organization

A picture of ADONIS framework presentation of the language is showed in the following (Figure 5-10).

Figure 5-10. ADONIS Worksheet example
5.4 PMOD

5.4.1 Description

PMOD [6] stays for Process MODelling. It is a diagrammatic notation defined to describe rigorously software processes. It is the result of a study project sponsored by ESA European Space Agency early in 1995 (Contract No. 11530/95/NL/JG). And further developed in a second contract (ECSS-PMOD Study ESTEC Contract No.1278/98/NL/PA). The intention was to describe formally the ESA software development standards such as ESA PSS-05 and later ECSS E40. INTECS played a major role (prime contractor) in its first definition. PMOD was also presented at:


A rigorously defined process was seen as a way to unambiguously define new ESA software development standards and also, in the long run, used as basis for process “enactment” tools. Enactment was used to indicate tools capable to drive and monitor that the process actually executed by project staff remained compliant with the defined process.

Though the notation was mainly fit to describe software processes it is general enough to describe actually any kind of processes, including manufacturing processes as presented in the magazine:


Basically PMOD originates from a data flow approach (inspired by SADT/IDEFO) which was already largely used to describe processes. A process is a transformation of input into output, articulated as a set of interconnected activities and data flows was an effective conceptual way to represent that.

Below are briefly illustrated the major concepts supported by PMOD.

A basic process unit is represented as follows:

![Activity Diagram]

It is then possible to decorate further this basic block with information related to:

- The **role** intended to execute that activity. Essentially role is intended with set of skills and level of independence where required.

- The **normative input** (e.g. standards, rules, guidelines, etc.) that are not real input to the activities but regulate “how” the activity has to be performed.

- A triggering event (a **pre-condition**). Usually an activity can start as soon as at least one input is available (may be in draft form), but sometimes there is the need to wait some specific events (e.g. a Kick-Off meeting).

- Supporting **tools**, libraries, good examples, etc.
Basic process units can be combined in a network of interconnected activities and grouped in a sort of **hierarchical structure** that can be zoomed-in and out at will.

So far so good. The illustrated notation is just a small variation of SADT/IDEFO maintaining all its expressive power slightly adapted to describe processes (e.g. role indication).

It was soon realized that in development processes we have to manage two kinds of activities:

**Constructive activities** that starting from some input produce an output providing added value. A typical example is the production of an architectural design (output) starting from a requirements specification (input).

**Verification activities**, where a given input is analysed with the sole purpose of detecting error, inconsistency, incompatibilities. The output is mainly a PASS/NO PASS. They act as **quality gates**.

The main purpose of a quality gate is to identify potential defects and stop the dataflow waiting for corrections of those defects. Often these verification activities provide output as “verification Reports” (e.g. test reports). The diagrammatic representation of these quality gate are exemplified below.
The notation intentionally does not express what to do in case the quality gate is not passed. This is intentional to avoid an overwhelming specification of corrective cycles. The notation assumes as implicit the following general rule:

Whenever a quality gate is not passed/crossed by a work-product an identified CCB (Change Control Board) will examine the nature of detected defects and re-start the process at the earliest stage where the defects are originated.

In principle if you find a bug late during running of executable code, and the analysis of the bug reveals that the problem is due to a specification error, the whole process has to be resumed as back as the requirements specification activity. This implicit resuming mechanism is called process backtracking (i.e. going back into its own steps). Backtracking has the effect of invalidating all previously generated work-product after the point of resumption, and new versions of work-products are generated.

The process description allows that any process may start as soon as some draft input is available. However an activity cannot terminate until the input is fully available. However it can produce a draft output. This mechanism of draft-in, draft-out is extremely useful as it allows to stress concurrency among activities, but a reconciliation is needed once the final input arrives and replace the draft input used instead.

### 5.4.2 Benefits and drawbacks

**Benefits**

The diagrammatic notation is simple. It can be grasped in few hours and starting describing processes. By the implicit backtracking mechanisms most processes are simplified by avoiding all return paths in case of defects. What is represented is just the “main correct flow”.

The notation does show “logical dependencies” between activities but do not force specific sequencing or timing.

The graphical notation is so simple that no specific tools are needed (Visio or PowerPoint were found adequate).

**Drawbacks**

The notation is known and restricted within the European Space Agency community and has little visibility outside. For large process models more elaborated tool support is needed.
5.4.3 Usability/applicability within the OPENCOSS development and certification approach

The origin of PMOD within the European Space agency made it a valuable candidate for the use in an European funded project leveraging from other European investments. Process models from PSS-05 and ECSS are already available. They implement a rather traditional process in compliance with ISO 12207 standard life cycle processes. High usability makes also it as a strong candidate.

5.4.4 Supporting tool and constraints

There is no specific tool support. The graphical notation is simple enough that process models can be defined using Visio or PowerPoint’s. However for large process models, it is felt the need of a more mature tool capable for instance to identify and highlight all activities performed by a give role (view-point). Or all input work-products affecting directly or indirectly a given output (e.g. for impact analysis).

5.5 Petrinets

5.5.1 Description

Petri nets [7] are a special form of directed graph; a bipartite graph. This means that there are basically two sets of nodes. In case of petri nets there are places (circular nodes) and transitions (square nodes). The bipartite graph now dictates that the start point is in the place-nodes and the end point in the transitions.

In contrast to many other graphs, petri nets have a strong semantic formalism. This makes them ideal for automatic reasoning and automatically detecting flaws. Since in petri nets there is no central control explicitly present, a typical application of petri nets is in the (massive) concurrent hardware systems.

5.5.1.1 Basic net: process steps, places, and transitions.

Models express the flow structure of processes. The simplest kind of model only uses three constructs and is called a basic net.

The most important modelling unit represents a process step. It appears in the diagram as a green square. Process steps form flows, and the process flow structure indicate that not every step can occur at every moment: they are subject to certain conditions. In petri nets, these conditions are made explicit. The squares represent actions, processes or events; generally speaking, transitions between states of affairs in the world. See Figure 5-11.
Figure 5-11: A net representing the states of a car

The circles represent conditions, and together, the possible states of affairs as far as they are relevant to the process flow of the process being described. They are known as places.

In Petri nets, arcs always run between transitions and places: it is illegal to connect two transitions or two places directly.

In practice we often see graphs or nets in which all branching happens on places: such models express alternative paths, but no parallelism or resource contention. A net with this property is called a state machine. For them, it makes sense to always omit the transitions and connect the places directly; this is in fact how state machines are usually drawn.

5.5.1.2 Tokens and markings

A state of affairs is represented by marking the conditions that hold. The dots are known as tokens; a distribution of tokens over places is a marking. See Figure 5-12.

Figure 5-12: tokens and markings

5.5.1.3 Process execution in basic nets

Process execution is defined by a single rule: a transition can occur whenever all places that have arcs to it contain a token; its occurrence will remove a token from each of them, and put a token in each of the places to which it has an arc. For example, in Figure 5-12, fuel the car can only occur when car out of fuel is marked with a token, and when it occurs, the token moves into car full of fuel. As soon as the token arrives there, car is driven around becomes possible; when it occurs, the token moves back into car out of fuel. So
we see the model is not very realistic: a car cannot be driven without running out of fuel, and it cannot be fuelled before it has run out.

5.5.2 Benefits and drawbacks

Petri nets are very expressive and they help us understand the behavioural aspects of a system. They constitute a bottom-up approach (appropriate for agile development) and can easily be composed into larger nets. Petri net engines provide extensive analytic capabilities, and they can do so even very early in a project. They also allow for recognizing starvation (conflict in which one transition always fires), dead lock, and mutual exclusion.

The down side of petri nets is that they often force a close look at the system at hand; they do not graphically scale up well. Other draw-backs, like that they are not event-driven, are solved in special event-driven versions of petri nets.

5.5.3 Usability/applicability within the OPENCOSS development and certification approach

Petri nets support developing a semantically unambiguous workflow. This is a major advantage in reuse and attempts to relate one workflow to another.

5.5.4 Supporting tool and constraints

At the Technical University of Eindhoven, YASPER (Yet Another Smart Process EditoR) has been developed to offer development of petri nets both in its pure form as well as in a more user friendly way, where common patterns are represented in different symbols.

5.6 Declarative workflows

5.6.1 Description

The process-aware information systems should both support business processes as well as provide flexibility. The traditional approaches, however, use procedural process models to offer good process support, but without much flexibility. In contrast with these approaches, the declarative approach assists in balancing between flexibility and support.

Figure 5-13 represents the classical process model in “inside out” style and the constraint based model in “outside in” style. This comparison shows specifically, by defining a limited set of constraints, the declarative workflows offer a more flexible process design and execution.
The declarative workflows [8] use a limited set of constraints in order to specify the execution procedure; these are the constraint based models. It means that any execution is possible as long as it does not violate constraints. Moreover, In [Ref], they demonstrate the declarative approaches by implementing Declare framework, which covers three different flexibility mechanisms: defer, change and deviate.

- **Defer** ("decide to decide later"): Declare facilitates flexibility by the possibility to easily defer choices to run time.
- **Change** ("decide to change model"): Declare allows for instance change at run time to support unforeseen situations or changed circumstances.
- **Deviate** ("decide to ignore model"): With optional constraints, Declare allows the designer to set a warning level to indicate the severity of the deviation.

### 5.6.2 Benefits and drawbacks

As mentioned earlier, the declarative workflow approach supports more flexibility than traditional. For general users, it provides the flexibility in choosing the order of steps rather than to be confined to an unnecessary strict process definition. Furthermore, for the company staffs, it offers more flexibility to apply staffs capacity to the necessary system development/quality assurance efforts.

However, the constraint-based approach is not quite suitable for processes of a strict procedural nature; As shown in Figure 1, in constraint-based approach, everything is allowed unless violates constraints, but in traditional approach, everything is forbidden unless explicitly specified. Therefore, for the highly procedural control-flow, the traditional approach is more suitable.

### 5.6.3 Usability/applicability within the OPENCOSS development and certification approach

In tailoring the processes prescribed in standards, companies/manufacturers often apply a more flexible approach. Both manufacturers and regulators allow for this flexibility, and it offers a much better flexibility in assigning capacity to the required system development and safety demonstration efforts. Declarative workflows offer to introduce this flexibility in a specified way, focussing on the real constraints in the safety demonstration process.
5.6.4 Supporting tool and constraints

As mentioned before, the concrete framework of declarative workflows is called Declare, which is a constraint-based WFMS (Work Flow Management System) and provide for multiple declarative languages (DecSerFlow, ConDec, etc.). The Declare system has three basic components: Designer, Framework and Worklist, Figure 5-14 shows the architecture of Declare. Designer component supports for designing and verifying the process model by creating constraint templates. The Framework component manages the execution of instances of process models, while the Worklist component is used to access active instances or execute activities in them.

![Figure 5-14 The architecture of Declare](image)
6 Supporting tools

6.1 Lifecycle supporting tools

6.1.1 Qualifying Machine

6.1.1.1 Description

Overall description

The Qualifying Machine (QM) [9] is an AdaCore project currently under development. Its goal is to ease the development, maintenance and reuse of certification material by providing many means to lighten some certification aspects, and in particular by providing an ‘as-automated-as-possible’ traceability between artefacts of different nature, documentation generation, and a framework for automating the artefacts generation.

![Diagram](image)

**Figure 6-1 The Qualifying Machine, Overall view**

Agile methods

The rational behind this focus on the automated traceability is the difficulty to apply Agile methods with certification activities, in particular because Agile Methods has a focus on constant refactoring ability at every step of the V cycle, and traceability – when performed by hand – is just a nightmare to handle in this context. Another challenge is to mimic the ‘continuous integration’ principle formalized by Agile Methods, and translate it as ‘continuous certification’ principle: at every step of the development project, we should be able to have a software that is in a certifiable state. This means that, at each iteration, all artefacts are coherent, all tests are passed, and all documents can be delivered. This is helped by automating the generation of artefacts (using Verification & Validation tools for example) as much as possible, automating the traceability as much as possible (and thus provide on-the-fly impact analysis), automating the documentation generation from various artefacts in various formats.
No Database aspect
Most software life-cycle supporting tools use database to store artefacts and traceability links. One of the main drawbacks of using databases in this context is the difficulty to keep it synchronized with the actual certification data (source code, test results, word documents, etc.). This is particularly true when the data is branched with a CMS, modified, or re-used from projects to projects. To ease reuse and branching, as well as interaction with external tools, the QM directly manipulates the source artefacts: they can be parts of a Document, standalone files, directories, databases or parts of a model. This handling is done without copying the content to a database, to prevent potential discrepancies between the actual data and the one managed by the QM, but via accessors that will directly extract the information from the original data.

Process handling
Contrary to traditional workflow supporting tools, the QM does not take as input a full workflow description, that has most of the time the drawback of fixing a waterfall development model, but instead put a stress on the constraints that apply to this workflow. As an example, the QM’s input model may define that test cases should be written from low level requirements. This implies that the impact analysis will show all test cases whose attached requirement has been modified since the test case was written. This of course does not imply that all LLRs should be written at once, and only afterwards the test cases, but instead that you can implement you software using small iterations, while making sure that the project remains fully coherent.
6.1.1.2 Benefits and drawbacks

The main drawback is that the QM is still under development, and is not yet feature-complete. That’s also an advantage as it can still be tailored and enhanced for its use by WP7.

6.1.1.3 Usability/applicability within the OPENCROSS development and certification approach

The Qualifying Machine can be used as a basis for the Work Package 7 in the following aspects:

- **Transparent certification**: providing metrics on the process, knowing the process constraints, and its current state. Knowing the set of artefacts of a given project, as well as the links that exist between those artefacts should allow OPENCROSS to define and calculate the new metrics we need.
- **Compliance-Aware Development process**: the traceability features of the QM as well as its on-the-fly impact analysis should allow the developers to make sure that the constraints that exist on the process - due to safety constraints or just compliance to standard constraints - are satisfied.

6.1.2 TOPCASED

6.1.2.1 Description

TOPCASED [10] is an integrated System/Software engineering toolkit compliant with the requirements of critical and embedded applications. It covers the stages from requirements analysis to implementation, as well as some transversal activities like anomaly management, version control, and requirements traceability.

TOPCASED is strongly model-oriented: not only TOPCASED provides model editors, model checkers and model transformations, but is also itself based on modelling and code generation.

TOPCASED is a meta-tool: you can develop your own graphical editors and model transformations using TOPCASED.

This description will not cover the modelling tools brought by TOPCASED themselves, but will focus on the features that can contribute to the quality of the product development it is applied to, that is to say:

- Requirements management
- Collaborative work
- Standardization
- Traceability handling

Requirements handling

TOPCASED allows importing external requirements to create a requirement model from a document. Several requirement importers exist, that are used in conjunction with style definitions, column selections and regular expressions. In particular, TOPCASED offers the possibility to import requirements from Microsoft Office documents (docx and xlsx documents), OpenOffice documents (odt and ods documents) or DOORS.

Traceability handling

Once the requirements are imported, TOPCASED allows the user to explicitly draw traceability links between those requirements and a model. The tool shows several metrics to benefit from those traceability links:

- A numerical value of the covered requirements percentage
- The distribution of the requirements depending on their state (covered/uncovered, has problems or is not covered anymore due to a model modification)

This traceability handling faces however many of the issues inherent to this category of tools: it is not resistant to the evolution of the client needs: if a requirement is modified, then the traceability link is not used to show that the model might require improvements in this area.
Main other features

Collaborative work:
TOPCASED allows splitting of the model so that several people can work simultaneously on the same project. It also provides merging capabilities so that conflicts emerging from the simultaneous modification of a given model by several people can be resolved.

Documentation generation:
As TOPCASED has access to the requirements and their resulting UML models, it can access all the information they own the same way. Thanks to this, this information can therefore be used in several ways, one of them being documentation generation.
Documentation generation is based on template documents, containing fixed parts (style, images, some textual parts), and a dynamic part that can access model data and manipulate them.

Scripting, Model simulation, batch generation of traceability matrix:
TOPCASED also provides many other tools related to the “Continuous integration” processes, such as batch generation of traceability matrix, model simulation, scripting and so on.

6.1.2.2 Benefits and Drawbacks
TOPCASED is the result of a French project financed by the “Aerospace Valley” competitiveness cluster in Toulouse. As such, it has pretty strong support in the aerospace industry. Moreover, the community around it is pretty large and the tool suite is well supported.
The main drawback of TOPCASED in regard to the goals of Work Package 7 is its only focus on model handling, and so its lack of handling of other certification artifacts (source code, unit tests, safety cases and so on). Another drawback is its lack of impact analysis feature.

6.1.2.3 Usability/applicability within the OPENCOSS development and certification approach
If not the tool itself, some of the principles that are behind this tool could be useful to the OPENCOSS project, and to WP7 in particular:
- The functional coverage metric is one of the metric that should be handled by OPENCOSS
- If not the only traceability link that should be handled by WP7, being able to link requirements and models is something that is also very desirable.
- The import functionalities of TOPCASED also show that it’s possible to extract artefacts from well formatted Word, Exel, or OpenDocument documents.

6.1.3 OSEE (Open System Engineering Environment)

6.1.3.1 Description
The Open System Engineering Environment [11] is an integrated, extensible tool environment for large engineering projects. It provides a tightly integrated environment supporting lean principles across a product’s full life-cycle in the context of an overall systems engineering approach. OSEE's feature set includes configuration management, requirements management, testing, validation, and project management.
OSEE is composed of several elements, working together:

- the Framework itself, allowing the communication and collaboration between the below elements. The framework provides the Artefact handling structures, handles the underlying database, and provides branching, configuration management capabilities. It’s one of the main improvements brought to Eclipse: binding different applications at the data level, not only the GUI level.
- the ATS (Action Tracking System) is responsible for tracking changes throughout the project’s life cycle, and performs impact analysis.
- the OTE (OSEE Testing Environment) provides a framework for the testing of complex software systems. It is tightly integrated into the OSEE to provide bidirectional traceability between software requirements, application code and tests.

6.1.3.2 Benefits and drawbacks
OSEE is a now pretty mature, and is backed up by Boeing, so really fits the certification objectives of the avionics domain. It allows the integration of development artefacts (the source code, belonging potentially to another pure development project in Eclipse) and certification artefacts. It also supports continuous testing in contexts typical to the avionics domain.

One of its main drawbacks is its lack of support for composition/reuse of artefacts: all artefacts being handled by a single database, they cannot be easily extracted together with their traceability links. An additional drawback is its Eclipse-only approach. There is no easy way to seamlessly integrate the tool with some external requirement editor, IDE or Eclipse non-supported tool chain.

Another potential issue with OSEE is its avionics domain specificity. Its adaptation to other domains such as the automotive or railway domains is not foreseen, and might require some major work.

6.1.3.3 Usability/applicability within the OPENCOSS development and certification approach
OSEE tries to solve many of the issues faced by development teams working in a certification environment. In particular, it brings together certification and development artefacts, in a way similar to what the
OPENCOSS project envisions. However if we want to reuse or adapt OSEE in the context of OPENCOSS, we need to take into account its monolith structure (Eclipse-only solution).

### 6.1.4 Concerto

#### 6.1.4.1 Description
Parasoft Concerto [12] is a complete Application Lifecycle Management tool. It facilitates management of requirements, defects, planning of project versions/iterations, scheduling work tasks and distributing them among team members. Finally it supports quality assurance of the product. The below figure presents how Concerto fits into existing software development infrastructure. The tool integrates with third-party Requirement Management Systems (RMSs) (e.g. DOORS, HP QC) and also provides its own RMS, it integrates with third-party Bug Tracking Systems (BTSs) and also provides its own BTS, integrates with Source Control Systems and collects automated tests results (Unit Tests and Coding Standards checks performed by Parasoft language tools). Concerto platform can be used by entire development team including requirement analysts, project managers, developers, testers.

![Concerto Diagram](image)

**Figure 6-5. Concerto and its interaction with external environment**

From OPENCOSS perspective the following group of functionalities could be of interest:
Requirements Management
Requirement Management System, with support for hierarchical structure and requirement modification history tracking.
Alternatively, Concerto can connect to external RMSs (DOORS, HP QC), synchronize requirements from there and facilitate traceability features (described below) for those requirements.

Requirements Traceability
For requirements stored (either synchronized from external RMS or stored originally in Concerto) the following traceability information is maintained:

Requirements to Tests traceability.
The tests associated with requirements can be:
1. Automated Tests sent from Parasoft Tools.
   For example: unit tests results sent from C++test, .Test, Jtest or automated functional tests sent from SOAtest
2. Manual Tests - textual test case scenarios which can be defined and run in order to validate the implemented/completed requirement

Requirements to Code traceability.
Specific file revisions can be associated with requirements. This way the information about lines of code and developers who are responsible for specific requirement implementation is preserved.

Requirements to Code Review traceability.
Parasoft platform facilitates Code Review (peer review) of source code. Having requirement-to-code traceability, Concerto provides Requirements - Code Review reports. This functionality may facilitate the possible following policy in the project: all the source code which is associated with an implemented requirement should be peer-reviewed and no issues found should be left open.

Requirements to Defects traceability.
When a defect is found in an implemented requirement and reported in Bug Tracking System, it can be associated with a specific requirement.

Requirement Modification history.
All the changes done to requirement are monitored and saved in a requirement change log.

Figure 6-6. Specific requirement traceability information
Project planning, collaboration and workflow

Concerto is an Application Lifecycle Management system thus facilitates various pieces of functionality for project planning and workflow enforcement.

Work planning and task distribution.
Project Manager can plan his project schedule in Concerto, divide requirements into work tasks, and assign the tasks to project milestones and to team members. Multiple features line Agile/Scrum burn-down charts, project end date calculation are present. As far as workflow features are concerned, the following features can be mentioned:

- When work tasks are assigned to owners, they are automatically notified about it by: email being sent, team members can import their tasks directly to their IDEs (Eclipse, Visual Studio) or see them in web interface.
- When work task is changed, all the interested parties (observers of specific item) are notified
- When Unit Test, Code Review or Coding Standard issue is detected, the team member which is responsible for the problem is notified and the items falls in his “to do” list

Requirement Test Status.
As Concerto keeps track of tests associated with requirements, it offers various reports presenting Requirement Test status, providing transparency of quality of requirements implementation, during implementation phase. Users can see which requirements have test scenarios defined, which of the tests scenarios were run and what was the status, which unit tests were run and what was the status - for each requirement in the project.

Requirement/Scenario review workflow
Requirement or Scenario items have “Status” property whose values can be customized (example values: New, Draft, To Review, Verified, Frozen, etc.). It can be configured which status values from-to transitions are allowed, or who is notified about specific value change.
This enforces the review workflow among project team members.

6.1.4.2 Benefits and drawbacks
Parasoft Concerto is a commercial product facilitating application lifecycle management. Its target is much wider then providing traceability and safety evidence - it also can be used for release planning, work progress monitoring (e.g. in Agile methodology), visualizing of test results from Parasoft automated test tools, etc.
The tool’s drawback, from Work Package 7 perspective, might be that Concerto is not aware of any specific safety certification or standard. If also does not store all the traceability which could be of OPENCOSS interest (e.g. safety cases).

6.1.4.3 Usability/applicability within the OPENCOSS development and certification approach
Some of the ideas could be useful for OPENCOSS project:

- Ability to integrate with third-party tools
- Artefact’s observers being notified upon artefact specific changes
- Traceability from requirement to tests, source code, code review etc.
- Reports presenting requirements traceability info from different angles.
6.1.5 FRASR

6.1.5.1 Description

Recall that OPENCOSs aims at providing efficient and affordable certification of safety according to internationally accepted standards. As argued above traceability between different software artefacts is essential for certification.

Similarly to the Qualifying Machine, TraceVis [13] and FRASR [14], two prototype tools that recently have been developed at Eindhoven University of Technology, aim at facilitating traceability. TraceVis is, as suggested by the name, is a generic traceability visualizer. Specifically, TraceVis has been designed to visualize a series of hierarchies (e.g., requirements documents, software components, tests or test results) and relations between them (e.g., software component implements a requirement or is covered by a test). Recently TraceVis has been applied to fraud detection in financial transactions and software model transformations.

Main advantages of TraceVis include its versatility and scalability. Versatility means that TraceVis can be applied to different domains, different artefacts and different aspects of safety assessment. The basic visualization technique, hierarchical edge bundles, has been specifically developed with scalability in mind. Moreover, TraceVis is a viewer and as such it is agnostic of what kind of data is visualized, i.e., TraceVis can be used in combination with any tool capable of detecting, traceability links.

Main shortcomings of the current version of TraceVis pertain to the fact that it is essentially a viewer, i.e., it is limited in the kinds of user interaction supported. These limitations will be addressed in an on-going TraceVis extension project.

Traditional approaches are limited to storage of traceability links provided by the user. As opposed to them the FRASR+ProM combination is capable of inferring such links, and therefore complementing the functionality of TraceVis. FRASR (FRamework for Analyzing Software Repositories) extracts information from multiple software repositories including version control systems, mail archives and bug trackers. Next, FRASR identifies related information fragments originating from different software repositories, e.g., e-mails sent, bugs reported and commits performed by the same person. Finally, these information fragments are combined into a log file. The log file is analyzed by ProM, a process mining tool. Process mining is a combination of techniques designed for analysis, model extraction and conformance checking of log files.

Traceability links created by the FRASR+ProM combination are twofold. First, identification of related information fragments carried out by FRASR can be seen as a form of traceability links identification. Indeed, in this way we can relate, e.g., bugs reported and code commits attempting to resolve these bugs. Second, during the process mining step carried out by ProM different activities recorded in the log are associated by means of temporal relations: e.g., activity A always precedes activity B. This kind of relations allows one to trace what steps lead to an (undesirable) result. Therefore, both the FRASR step and the ProM step provide traceability links required for certification.

Summarizing the discussion above TraceVis and FRASR+ProM can contribute to an improved certification or safety assessment transparency. Indeed, certification transparency can only be achieved if the user has complete and profound understanding of traceability relations between software artefacts involved in the certification.
6.2 Process description tools

6.2.1 Eclipse Process Framework

6.2.1.1 Description

As stated on Eclipse’s website, the Eclipse Process Framework (EPF) [14] aims at producing a customizable software process engineering framework, with exemplary process content and tools, supporting a broad variety of project types and development styles. The Process Framework Project has two goals:

- to provide an extensible framework and exemplary tools for software process engineering - method and process authoring, library management, configuring and publishing a process,
- to provide exemplary and extensible process content for a range of software development and management processes supporting iterative, agile, and incremental development, and applicable to a broad set of development platforms and applications.

The Eclipse Process Framework Composer (EPF Composer) is proposed as an exemplary tool. OpenUP, XP and Scrum are proposed as exemplary process frameworks. This tool and these frameworks are available as free downloads under the Eclipse Public license.

EPF Composer enables process development and deployment by either:

- tailoring existing contents of available process frameworks like OpenUP, or
- creating contents from scratch.

Data managed by EPF Composer are split into two main categories:

- method contents,
- processes.

Method contents are descriptions independent from lifecycles. They are constituted by “work products”, “roles”, “tasks” and “guidance”. Work products, roles and tasks correspond to “what”, “by whom” and “how”. Guidance comprises guidelines, templates, checklists, etc. Method contents are organized in categories, among which we find “disciplines”. For example, an “Initiate project” method could be part of a “Project management” discipline.

Processes define the selected lifecycle. They take method contents as building blocks, relating them in semi-ordered sequences. This enables both parallelism between methods and multiple uses of a given method. Method contents might be customized each time they appear in a process. For example, not all the work products of a method serve as inputs or outputs in every instance of this method. The same is true for roles and tasks. For example, some steps may be unnecessary.

More precisely, processes are constituted by a breakdown of nested activities and by milestones. Activities can be phases or iterations. Task descriptors, role descriptors and work product descriptors enable the customization. Entry states and exit states are associated to each work product relevant for an activity, defining beginning and end criteria for that activity.

EPF Composer generates a website from the method contents and processes definitions. If the involved stakeholders have online access to this website, deployment is straightforward. After any modifications leading to a new baseline, the website gets updated.

6.2.1.2 Benefits and drawbacks

EPF is powerful. It may appear complex to some users. Its flexibility and modularity make it suitable for descriptions ranging from high-level processes to detailed and complex processes. The EPF project claims to
seek conformity to OMG’s SPEM 2.0 standard (cf. section 5.1). Supported by the Eclipse Foundation and an active community, EPF is expected to be a long-lasting project.

EPF Composer is a process modelling and publishing tool. It is coupled with CVS to support the authors of process descriptions.

EPF Composer can export data in a format readable by Microsoft Project to provide the project manager with a WBS consistent with the process description. Apart from this, it does support the execution of the processes.

6.2.1.3 Usability/applicability within the OPENCOSS development and certification approach

Even if software development is highlighted in EPF’s project goals, EPF is restricted neither to the software context nor to development activities. It may be used in contexts like system or hardware engineering or certification.
7 Gaps between the Baseline Survey and the State of the Art

Based on the results of the baseline survey and the review of the state of the art, this section analyses the gaps that can be found between them. Its purpose is to discover differences and determine (1) what are they? (2) Why and how they can affect the WP, and (3) whether they can or should be addressed in WP7 (and, if yes, how).

The gaps have been divided into the following categories and they are presented and discussed in the following subsections:

- Transparent certification process
- Compliance-aware development process approaches
- Process description formalisms
- Supporting tools

7.1.1 Gaps in transparent certification process

Based on the baseline survey we have gathered all the results and we have compared what exists with respect to what is performed by organisations. The first aspect we need to clarify is what factors are the most relevant for this Transparent Certification process. The following figure shows the most important factors extracted from the survey results.

![Factors important for the transparency of the certification process (percentage)](image)

Figure 7-1. The most relevant factors related to transparency certification process

These factors have an evident impact in this kind of critical software systems, and into the entire product lifecycle. In fact several actors are involved in the certification process and it is not just a matter of auditors or quality assurance managers. As it is reflected in the following figure system end users, integrators and suppliers are also involved in the safety certification processes.
The main stakeholders and beneficiaries in the safety certification process

- OEMs (Original Equipment Manufacturers)
- System assessors and certifiers
- System Suppliers
- System Integrators
- System end-users

**Figure 7-2. The main stakeholders and beneficiaries in the safety certification process**

The main conclusions of the comparison are as follows:

- Understandability, Traceability, Visibility and Completeness are the most relevant factors for transparent certification processes.
- Several stakeholders are involved in the transparent certification process. This means that the resulting OPENCOSS platform should provide a way to manage all this kind of users and roles. In fact each of these roles is not involved in the same level during the certification lifecycle as the other roles. provide input to developers on compliance awareness
- metrics on safety assurance activities: e.g. costs on delta certification according to the delta certification activities

### 7.1.2 Gaps in compliance-aware development process approaches

Some of the interesting aspects revealed in this survey it is the relationship between development processes and the certification processes. The main conclusions are the following ones:

- First of all partners and it is represented in the following figure they are willing to use agile approaches for development processes, but also to the certification process because they argue that the actual certification processes used are too rigid and not flexible to be adapted to each situation without removing discipline and thoroughness.
Agile approaches

No: 13.33%

Yes: 86.67%

Figure 7-3. Agile approaches are considered to be used

- Interwoven safety assurance and certification processes with development processes. The relationship between them is often mixed and blurred. In fact partners have difficulties to clearly identify which aspects are related to their development processes, which ones are defined for assuring safety and which ones are related to the certification processes. This would be necessary to clearly identify and separate these aspects in order to quantify how much time, efforts and so on take each of the above processes.

7.1.3 Gaps in process description formalisms

In order to define the certification processes we have analysed the most frequent process description formalisms. Thus we have extracted a set of tools and languages used by our partners. The following figure shows the use of graphical notations used for defining processes.
Based on these data we can conclude the following important conclusions:

1. **UML based process definition languages** (e.g., SPEM) are predominant in the OPENCOSS consortium. Therefore OPENCOSS solutions should be based on this kind of languages. The selected language will be used not only for describing development processes but also certification processes in order to make explicit the whole certification process.

2. There are still organisations not using languages for describing processes. This aspect makes difficult a transparent certification process because they do not have a clear understanding of these certification steps, and how they can impact into their daily activities.

### 7.1.4 Gaps in supporting tools

This is a weak area where there is no consensus on what kind of tool is used for certification processes. In fact they listed the following type of tools:

- YAWL, Systemation’s AgilePoint [16]
- Goedelworks [17]
- TECNALIA Process Factory Studio [18]
- Atego Process Director [19] for process authoring, dissemination and data collection.
- ADONIS [20]

This is really an opportunity for OPENCOSS because it is one of the key aspects and results of this project.
8 Related projects

8.1 FP Projects

8.1.1 FACIT-SME

8.1.1.1 Overview
FACIT-SME (Facilitate IT-providing SMEs by Operation-related Models and Methods) is a European (FP7 - SME-2008-2) [71] research project targeting facilitated the use of Software Engineering (SE) methods and to systematize their application in IT SMEs. It will provide methods for efficient and affordable certification of these processes according to internationally accepted standards. Furthermore, it will help the SMEs to share best practices, tools and experiences with development partners and customers. ICT SME communities (organized by associations) will experience significant benefit through the exchange of recent knowledge and best practises.

8.1.1.2 Objectives and Benefits
FACIT-SME’s objective is to promote the competitive position of the European IT industry. It will enable SME Associations to provide operational support to their IT SME members in order to become more competitive and efficient, by providing operational applicable Best Practice as well as methods and tools. The project is not intending to invent new methodologies and tools for software engineering but to facilitate their application at SME level.

SME associations will improve their attractiveness to their existing and new partners by:
- Providing operational reference knowledge and services to support the operational business of partners
- Enable structured knowledge exchange and transfer among the partners SMEs will be supported through:
  - Seamless application of suitable model-based methods and tools to support software engineering companies
  - Provision of SME-adapted management systems to increase quality and operational excellence without much extra effort
  - Ensure better market position through the support of multi-standard certification
  - Ensure community-based information sharing and efficient collaboration using Web 2.0 technologies
8.1.3 Potential exchanges with OPENCOSS

Basically the relationship between FACIT-SME and OPENCOSS it is based on the support of multi-standard certification. This context is similar to OPENCOSS where different reference standards and technologies should be fulfilled at the same time during the definition and execution phases. In fact FACIT-SME works on the needs of SMEs with respect to the compliance of several quality models. In addition it provides some tools and approaches for dealing with multiple standards and reference models.

8.2 ARTEMIS Projects

8.2.1 SafeCer

8.2.1.1 Overview

SafeCer (Safety Certification of Software-Intensive Systems with Reusable Components) is a European (Artemis) [72] research project targeting increased efficiency of safety-relevant embedded systems. The industrial domains targeted are within automotive and construction equipment, avionics and rail. SafeCer will also develop certification guidelines and a training example for other domains. The SafeCer project is divided into two projects: a pilot project, pSafeCer, and the demonstration and implementation project, nSafeCer. SafeCer as a whole started April 2011, and should end April 2015. It involves 31 partners.

8.2.1.2 Objectives and Benefits

SafeCer’s objective is to promote the composition and reuse of ‘certification components’, inside or across domains. This is performed via 4 main research activities:

- Definition of the Process Model (PM), used to describe actual certification process.
- Definition of a Component Model (CM), that defines additional properties to the certification process to allow component-based software development approach.
- Creation of the Certification Tool Framework (CTF), whose goal is to control verification and validation tools and extract the metadata contained in the artefacts produced by those tools
- Creation of the Certification Artefact Repository (CAR), whose goal is to store the metadata created by the CTF, handle the traceability links between those artefacts and provide impact analysis, as well as process compliance verification.
The relation between the Process Model, the Component Model, the Certification Tool Framework and the Certification Artefact Repository is shown in Figure 8-2.

8.2.1.3 Potential exchanges with OPENCOSS

SafeCer is very similar to OPENCOSS in terms of goals and means to achieve them. For example the Certification Tool Framework and the Certification Artefact Repository are very similar to OPENCOSS’ WP6 and WP7. As already detailed in the deliverable D6.1, it seems sensible to think that OPENCOSS and SafeCer could exchange results, so that both projects can benefit from each other.

8.2.2 CHESS

8.2.2.1 Overview

The CHESS Project [73], ending in April 2012, is partially funded by the Artemis Joint Undertaking. It sought to improve Model Driven Engineering practices and technologies to better address safety, reliability, performance, robustness and other extra-functional concerns while guaranteeing correctness of component development and composition for embedded systems.

8.2.2.2 Objectives and Benefits

The CHESS methodology for the development of critical real-time systems is based upon a model driven engineering approach, whereby the central artefacts of development are models rather than documents or code. It is a demanding approach that requires comprehensive tool support for all aspects of the development process.

In addition to being model based, the CHESS development process is component based, building on the important advances in component based engineering that occurred in the last ten years. Special emphasis is set on providing advanced support for the rich and precise specification of the extra-functional
characteristics of these components, especially in the dimensions of time predictability, isolation, transparency, dependability and safety.

The other distinguishing element of the CHESS methodology is the concept of separation of concerns. This concept is a long-known best practice for the successful development of complex systems, in which developers “divide and conquer” the problem by focusing separately on key aspects such as functionality, real-time behaviour, deployment – and even other stakeholder concerns such as economic costs – rather than by trying to capture all aspects simultaneously in one exceedingly complex big picture.

The methodological and technical means by which CHESS provides the developer with support for separation of concerns is the popular, well-accepted modelling concept of views, most prominently acknowledged in modern modelling languages such as UML. The CHESS Methodology associates a distinct view with a distinct concern pertinent to in system software modelling.

CHESS implements the principle of continuous verification that guarantees that the model is always consistent and up-to-date and that supports the iterative development. While incremental development is provided by its component-oriented approach, whereby systems are composed of components chosen from repositories.

The CHESS tool-chain guarantees the implementation of the correct by construction paradigm by supporting:

- the analysis and verification of extra-functional properties (predictability, dependability and security) in the component-based software system modelling and assembly
- the propagation of the results back to the model to assure guarantee of the extra-functional properties for the component compositionality and composability
- the consistent generation of code and deployment on the target platform.

### 8.2.2.3 Potential exchanges with OPENCOSS

The expertise of some partners gained in participating to the CHESS project and the methodological approach developed by the project can be valuable input to the compositional certification requirements and to the compositional conceptual framework of OPENCOSS.

### 8.2.3 RECOMP

#### 8.2.3.1 Overview

“Reduced Certification Costs Using Trusted Multi-core Platforms” (RECOMP) [74] is a European research project which aims at enabling cost-efficient (re-)certification of safety-critical and mixed-criticality, component based, multicore systems. Applications addressed are automotive, aerospace, industrial control systems, and lift and transportation systems.

The RECOMP project spans from April 2010 to March 2013. It is carried out by a consortium composed by 41 partners from 9 countries: 30 Companies (9 small, 4 medium and 17 non-SME) and 11 universities and research organisations. RECOMP is a European funded project from ARTEMIS JOINT UNDERTAKING (JU).

RECOMP will provide reference designs and platform architectures, together with the required design methods and tools. The aim of RECOMP is to define a European standard reference technology for mixed-criticality multi-core systems supported by the European tool vendors participating in RECOMP.

#### 8.2.3.2 Objectives and benefits

As stated on RECOMP’s website, the concrete objectives of the project are the following:

- Objective 1: Develop design methods and tools for certification and re-certification of mixed criticality applications on multi-core platforms.
- Objective 2: Develop hardware and middleware platforms based on application independent HW/SW mechanisms that enable safe multi-core virtualisation and core-to-core communication.
- Objective 3: Demonstrate reduced certification and re-certification costs of component-based systems implemented using the RECOMP methods and tools on the RECOMP platforms.
The participating and collaborating partners aim to achieve at least the following advantages:

- Lower total system cost: By being able to use multi-core the overall system cost is reduced, as the full application can be run on a single circuit board module that saves mass, volume and power.
- Lower cost of the certification of first version of the product: By using a certified computing platform the cost of certification will be reduced.
- Shorter time-to-market: The certified platform will reduce time required for the certification of the application
- Lower re-certification costs: Re-certification costs can be kept low since non-critical parts of the application can be updated without touching the safety critical parts of the application. In addition, different parts can be incrementally accepted due to the separation into different levels of criticality.
- Supplier-based development process: The possibility to develop pre-certified components will enable component suppliers to take on a bigger share of the product development share, distributing costs over several customers, and thus lower overall costs of the development process.

### 8.2.3.3 Potential exchanges with OPENCOSS

RECOMP’s technical work is divided in work packages as follows:

- **WP1**: Research drivers and Business Case,
- **WP2**: Design methods and tools,
- **WP3**: Trusted multi-core platform,
- **WP4**: Certification lifecycle issues,
- **WP5**: Demonstrators,
- **WP6**: Evaluation.

WP4 is of particular interest for the OPENCOSS project. The associated deliverables are:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Nature</th>
<th>Level</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4.1a.1</td>
<td>Overall Safety Requirements for Automotive/Industrial Control</td>
<td>Report</td>
<td>Restricted</td>
<td>T0+6</td>
</tr>
<tr>
<td>D4.1b.1</td>
<td>Safety Requirements for Aerospace Study Case</td>
<td>Report</td>
<td>Restricted</td>
<td>T0+6</td>
</tr>
<tr>
<td>D4.1a.2</td>
<td>Safety Certification Workshop IEC 61508</td>
<td>Other</td>
<td>Confidential</td>
<td>T0+4</td>
</tr>
<tr>
<td>D4.1b.2</td>
<td>DO-178B Workshop</td>
<td>Other</td>
<td>Confidential</td>
<td>T0+4</td>
</tr>
<tr>
<td>D4.2a.1</td>
<td>Standards Amendment targeting multiple processor safety systems (SMP, AMP)</td>
<td>Report</td>
<td>Restricted</td>
<td>T0+24</td>
</tr>
<tr>
<td>D4.2a.2</td>
<td>Development process amendments to achieve a robust and safe development</td>
<td>Report</td>
<td>Restricted</td>
<td>T0+24, T0+36</td>
</tr>
<tr>
<td>D4.2b.1</td>
<td>Recommendations for use of multi-core in certifiable applications for Avionics</td>
<td>Report</td>
<td>Restricted</td>
<td>T0+24, T0+36</td>
</tr>
<tr>
<td>D4.3a.1</td>
<td>Lesson Material of multiple processor Safety Systems Architectures (SMP, AMP)</td>
<td>Other</td>
<td>Public</td>
<td>T0+36</td>
</tr>
<tr>
<td>D4.3b.1</td>
<td>Migration Method from DO-178 to IEC 61508 certification (SMP, ASP)</td>
<td>Report</td>
<td>Public</td>
<td>T0+36</td>
</tr>
</tbody>
</table>

However, these deliverables are either not accessible due to their dissemination level or not available as of today.

Several deliverables from other work packages are already publicly available on RECOMP’s website, in draft or final versions, together with technical reports. Most of them tackle numerous subjects related to multicore multiprocessor architectures (hardware, firmware, RTOs, middleware, tools, scheduling, partitioning, synchronisation, determinism, resource sharing, fault tolerance, monitoring, virtualisation…). These documents are not directly related to the certification process and thus do not constitute useful inputs for OPENCOSS.

In D2.1 ("Component model specification"), an interesting description of certification in the industrial, aerospace and automotive domains is given. In particular, thoughts about “main obstacles for certifying safety-critical multi-core products” show that RECOMP and OPENCOSS share many findings regarding certification.

As a conclusion, exchanges between RECOMP and OPENCOSS might be very fruitful. Currently available information is not sufficient to determine whether and how this could happen practically.
9 Conclusions

Main challenges in safety assurance and certification from multiple domains

This document re-affirms the issue of the so-called “Babel of standards”. Despite the original intention of IEC 61508 to represent a single multi-domain safety standards from which other standards may be derived, reality is rather different. Even in those cases where they are really derivative of IEC 61508, the differences are buried in the text and it is not easy to identify them.

As a consequence, the certification process too is formulated in a number of different interpretations, ranging from the automotive domain that claims “we have no such concept as certification” up to the railway where government authorities culminate a complex certification process.

The notion of process itself is expressed differently among safety standards. While most of the standards rely on a classic waterfall model, the avionics standards explicitly states that no particular process is prescribed, as long as the process has the correct properties, contains the proper activities.

This proliferation of standards and approaches is only partly justified by the different needs of the different domains. It is instead largely due to historical reasons and the fact that these standard have emerged from different communities with no interest to share their approach with other. To some extent different standards also created a kind of entry barrier in a market sector, thus protecting the interest of those already inside that market.

As shown in the results of the survey, presented in the chapter 7, the OPENC OSS consortium is a great opportunity to define a common framework that help align the safety assurance practices in order to cope with those different standards. The partners involved in the project have a great experience in the development of safety-critical systems, for the Avionics, the Railway and Automotive domains.

This will allow us to focus on the main issues they are facing with their safety assessment, quality assurance, certification and development processes.

In particular, the survey showed one trend of the market: the agile methods - even if currently scarcely adopted by the safety-critical industries - are looked at closely and are bringing more and more interest.

Agile and Safety: toward an agile assessment?

Agile is probably the most visible progress in software engineering of the last decade. The many life cycle models such a waterfall, incremental, evolutionary, spiral, etc. have culminated in the so called IID Iterative Incremental Development with a time-boxed progress. The safety assessment shall be adapted to cope and be interwoven with such a new life cycle style. Even large corporations such as Thales have successfully reported (Toulouse ERTS 2012 Conference) of an avionic software DO178B level A, developed with Agile approach.

Can we start talking about an Agile Assessment? Yes, definitively. The assessment shall proceed along with the development process, by increments. We shall assure that we build a safe product from the very beginning. In principle, at each increment the product shall be and shall remain safe (avoid safety regressions). This may not be fully practicable achievable, but it is the goal to approximate as much as possible.

However applying Agile methods to safety assessment is not so trivial and should be considered carefully. In particular, the Agile Manifesto tries to lower as much as possible documentation, which is not an option in
safety standards, as documentation is the main output of the certification/safety assessment activities. Another issue concerns the constant refactoring that is highly recommended when using Agile methods. This refactoring makes traceability between artefacts difficult to keep updated.

How do we effectively achieve this is not the subject of this documents. The European Space Agency has proposed in a recent software engineering handbook that when iterations are very short, e.g. one week, it may be too expensive to perform full verification and validation activities that can be synchronized, e.g. every 4 iterations. The same approach could be proposed for assessments. It may be not economics to repeat a safety activity (e.g. a FMEA) at every iteration, and leave it after a fixed number of iterations.

For the purpose of clarification, in a sense what is emerging is that (using the SCRUM terminology) we may have something like:

a. Daily building (an increment of product running and tested)
b. Two-weeks “sprint” (an increment of product with all documentation up-to-date)
c. Two-months “assessment” (an increment of product demonstrated safe)

In order to achieve that it is fundamental to deploy a certification platform tightly interfacing the development repository leading to an incremental and evolutionary assessment that we can also call an “agile assessment”.

Process Notation and Process Enactment

The certification process shall adopt the same kind of notations used to describe development process. To some extent the certification process will result as an over-imposed process on top of an ordinary engineering basic process.

SPERM seems the most suitable, but still in a technology niche. BPMN is extremely popular and powerful. PMOD has good roots but it is not as popular and well supported by tools as BPMN is, which seems therefore the best starting point.

As for the enactment of the process (e.g. the ability to drive the humans in the proper execution of the process respecting dependencies and constraints), the approach proposed by the Qualifying Machine seems the most promising and has the advantage to be fully master by a project partner.

In a crude simplification, once upon a time we had only a Unix Makefile that, having instructed about dependencies of source files, arranged the compilations and linking accordingly. Now the qualifying machines extends this concept to all project artefacts (work products), not only source files, and extends not only to the compilation but to all project activities, including those activities requiring human intervention such as “execute a code inspection on module X, verify the document Y).

The engine is therefore driven by rules and dependencies, assuring that the process is always executed in the correct order, by the correct people, with no omissions.

However, in order to be coherent with the overall project policy toward open source and pave the basis for an effective exploitation of results, investigations are required on integrating the Qualifying Machines with open source platform such as TOPCASED (and of course OPENCOS). The open source license scheme has still to be investigated.

Assessing the safety assessment: toward a transparent certification process
Another issue identified by OPENCOSS is the frequent lack of transparency in the certification process: the visibility on the safety-related artefacts and process and its relationship with the development artefact is very often not so clear. This is an issue for certification, as well as safety managers responsible for the correct execution of the safety assessment and thus might impair the safety itself.

In order to provide this transparent certification, an integrated tool needs to be created and be made aware of all artefacts and activities of a given project, including both development and certification-related artefacts. It would then be able to provide the proper metrics and specific views on those artefacts and activities to better understand where the project is in terms of safety-assurance and certification process execution.
# 10 Abreviations and Definitions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTEMIS</td>
<td>Advanced Research &amp; Technology for EMbedded Intelligence and Systems</td>
</tr>
<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</td>
</tr>
<tr>
<td>SPEM</td>
<td>Software Process Engineerirng Modelling</td>
</tr>
<tr>
<td>QM</td>
<td>The Qualifying Machine</td>
</tr>
</tbody>
</table>
11 References

[14] www.frasr.org/description/
[17] www.altreonic.com
[31] http://manu40k.free.fr/AgilityAndLeanForAvionics1.pdf
[33] http://www.eetimes.com/design/embedded/4227764/The--Big-Thaw----An-Agile-Process-for-Software-Certification?-pageNumber=1
[34] http://www.exforsys.com/tutorials/testing/testing-for-agile-software-development.html
[40] ED-80/DO-254, Design Assurance Guidance for Airborne Electronic Hardware, EUROCAE/RTCA
[43] CENELEC EN 50126, Railway Applications: The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS), September 1999
[46] CENELEC EN 50159, Railway Applications: Communications, signalling and processing systems, March 2001
[47] IEC 61508 (all parts), Functional safety of electrical/electronic/programmable electronic safety-related systems
[52] ECSS-E-ST-40 Space engineering – Software
[53] ECSS-Q-ST-80 Space product assurance – Software product assurance
[54] ECSS-Q-ST-30 Space product assurance – Dependability
[56] FDA Food and Drug Administration Standards
[57] IEC 62304 (software life cycle + SILs e tailoring)
[58] ISO 13485 (Quality Systems Requirements, a kind of tailoring ISO 9001) (replaces ISO 46001) (considered fundamentally equivalent to FDA QSR Quality System Requirements)
[59] ISO 14969 (it provides a guidance to ISO 13485)
[60] GAMP 5
[61] GMP Good Manufacturing Practices
[62] CDRH Center for Devices and Radiological Health (many guidelines)
[63] ISO 14971 Risk management for medical devices
[64] EN 60601 1-4 Programmable electrical medical systems
[65] EN 60601 1-6 basic safety and essential performance - usability processes for medical device software
[66] TIR 32:2004 Medical Device Software Risk Management (=> IEC 80002-1)
[67] IEC 60812 FMEA
[68] European Council MDD (Medical Device Directive) 93/42/ECC Concerning Medical Devices
[69] ISO 25119-4:2010 Tractors and machinery for agriculture and forestry -- Safety-related parts of control systems -- Part 4: Production, operation, modification and supporting processes
[70] ISO 4254-1:1985 “Tractors and machinery for agriculture and forestry – Technical means for ensuring safety”
[73] http://chess-project.ning.com/
[74] http://atc.ugr.es/recomp/
[77] Model Based System Engineering at Lockheed Martin MS2 http://goo.gl/COV2S