A Tool-supported Model-based Method for Facilitating the EN50129-compliant Safety Approval Process

Faiz Ul Muram[1][0000-0001-6613-4149], Barbara Gallina[1][0000-0002-6952-1053], and Samina Kanwal[2]

[1] School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
`faiz.ul.muram|barbara.gallina@mdh.se`
[2] National University of Sciences and Technology
Islamabad, Pakistan
`saminakanwal5231@gmail.com`

Abstract. Compliance with the CENELEC series is mandatory during the planning of as well as development of railway systems. For compliance purposes, the creation of safety plans, which define safety-related activities and all other process elements relevant at the planning phase, is also needed. These plans are expected to be executed during the development phase. Specifically, EN 50129 defines the safety plan acceptance and approval process, where interactions between the applicant and the certification body are recommended: after the planning phase, to ensure the compliance between plans and standards, and after the development phase, to ensure the effective and not-deviating-unless-justified execution of plans. In this paper, we provide a tool-supported method for facilitating the safety approval processes/certification liaison processes. More specifically, the facilitation consists in guidance for modelling planned processes and the requirements listed in the standards in order to enable the automatic generation of baselines, post-planning processes and evidence models, needed during the execution phase and change impact tracking for manual monitoring of the compatibility between plans and their execution. The applicability of the proposed method is illustrated in the context of EN 50126-1 and EN 50129 standards.

Keywords: EN 50129 · EN 50126-1 · safety management · safety processes · regulatory compliance · safety plans · model transformation.

1 Introduction

In the context of railway systems engineering, the Comité Européen de Normalisation Electrotechnique (CENELEC) standard series defines a set of norms as well as a set of processes to be followed. Process planning is one of these processes, which involves development of safety plans, which define: the units of work (such as phases, activities, tasks), expected to be executed during the development; a set of methods to be used; work products to be taken as input or produced as output; involved roles, expected to take responsibility for the execution of the work. In avionics, DO-178C [19], the de-facto standard for airborne software development defines the *certification liaison* *process*, where the interactions (Stages of Involvement-SOI) between the applicant and the certification body are expected to take place throughout the life-cycle of a project. In particular, the first interaction (SOI#1) is expected to take place after the planning phase to ensure that plans are compliant with DO-178C objectives. The second interaction (SOI#2) is expected to take place after the execution phase to assess the project-specific implementation against the approved plans and the DO-178C requirements (i.e., to ensure that the activities to be undertaken are fully congruent). Similarly, in the context of railway systems, the EN 50129 standard [10] defines the reviews (safety acceptance and approval process), which shall be carried out at appropriate stages in the life-cycle. For the safety plan approval, a checklist of activities and items shall be produced in compliance with the CENELEC standard series. The review of the safety plan after each safety life-cycle phase is also recommended.

For getting the approval of safety plans from the certification body, the compliance between the safety plans and the CENELEC standard series requirements should be shown. Furthermore, for the getting the approval for the evidence produced during the development, compatibility between the executed process and the planned process should also be shown. Therefore, it is necessary to provide a reference between the safety plans and the CENELEC standard series requirements as well as a reference between the executed process (including the corresponding evidence) and the planned process. Managing manually such traceability is a tedious and challenging task because of the large amount of information, which needs to be handled. Also, since most of the exiting approaches used to document process models representing plans are natural language-based, the automation of such task is hindered. To facilitate the approval process and more specifically the automatic management of process-related compliance information, in this paper, we provide a novel tool-supported method, which consists of: guidance for modelling (in compliance with the Process Engineering Metamodel (SPEM) 2.0 [18], and more specifically with its reference implementation, implemented in EPF (Eclipse Process Framework) Composer[3]) safety plans and the requirements listed in the standards in order to enable the execution of our proposed model transformation for generating baselines, post-planning processes and evidence models (in compliance with Common Assurance and Certification Metamodel (CACM) [4], implemented in OpenCert[4], which enables the evolution and traceability of models during the development phase), needed during the systems development phase. The transformation is achieved by using Epsilon Transformation Language (ETL)[5]. Specifically, a set of ETL transformation rules are used to transform the CENELEC standard series requirements into the baseline models and diagrams; whereas the safety processes are transformed into the first-view of post-planning processed and evidence models. By automatically generating baseline and evidence models within an environment that supports traceability, this model transformation facilitates the compliance demonstration and thus the plan and its substantiation's approval. Moreover, once the modelling of the standards is completed, process engineers might dedicate their time to the manual production of portions of expected outputs/deliverables that strictly require human

[3] `https://www.eclipse.org/epf/`
[4] `https://www.polarsys.org/proposals/opencert`
[5] `https://www.eclipse.org/epsilon/doc/etl/`

intervention. The applicability of the proposed method is illustrated for EN 50126-compliant design specification [9] and EN 50129-compliant safety plan acceptance and approval process [10], focusing on the safety demonstration for a generic product (i.e. independent of application).

The rest of this paper is organized as follows: Section 2 presents essential background information. Section 3 describes the tool-supported model-based method for the transformation of standard compliant planned process models to baseline, post-planning process and evidence models. Section 4 illustrates the application of our approach for CENELEC EN 50126 and EN 50129 standards. Section 5 presents the related work. Finally, Section 6 concludes the paper and presents future research directions.

2 Background

This section recalls the background information on which the presented work is based: in particular, Section 2.1 recalls the necessary information regarding the CENELEC standard series. Section 2.2 presents the process modelling language used in this paper. Section 2.3 recalls basic information about EPF Composer. Section 2.4 recalls essential information about CACM metamodel and the OpenCert tool. Finally, Section 2.5 recalls basic information regarding model-driven engineering principles and techniques.

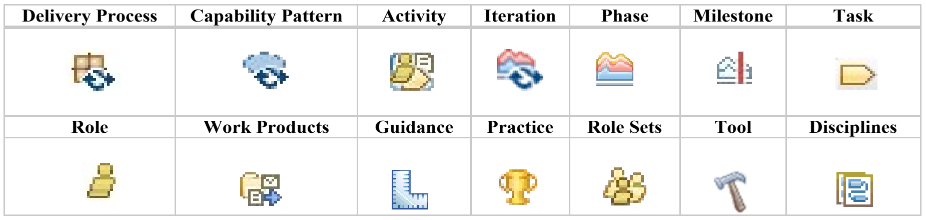
2.1 CENELEC Series

The CENELEC series is a set of standards, which contains requirements and recommendations concerning processes to be followed during the planning, development, deployment and maintenance of the railway systems. EN 50126-1 [9] is part of the CENELEC series. EN 50126-1 provides a fourteen-phase life-cycle process, known as the RAMS process, for developing railway systems by focusing on Reliability, Availability, Maintainability and Safety. The verification and validation activities take place throughout each phase of life-cycle process. In this paper, we limit our attention to Phase 6 (Design and Implementation). The main objective of this phase is to design the sub-systems and components in conformity with RAMS requirements. This phase includes general tasks (e.g., planning, design and development, design analysis and testing, implementation, and verification and validation) and the safety tasks (e.g., preparation and application of safety cases, and the justification of safety related decisions). The verification tasks associated with this phase include the verification of design and realisation of sub-systems and components against RAMS requirements, future life-cycle plans, competence of all personnel, methods, tools and techniques used in this phase, and verification of safety case design and application etc. Each task is associated with the expected output/deliverable or artefacts showing the evidences of requirements. EN 50129 [10], also part of the CENELEC series, defines the three conditions that shall be satisfied in order that a safety-related electronic railway system/sub-system/equipment can be accepted as adequately safe for its intended application. These three conditions are: 1) evidence of quality management (including quality planning and organisational structures) to be documented in the quality management report; 2) evidence of safety management, expected to be consistent to the RAMS process recommended in EN 50126-1 and expected to be documented in the Safety Management Report; and 3) evidence of functional and technical safety. This paper facilitates the satisfaction of the first two conditions.

2.2 Process Engineering Metamodel

SPEM 2.0 [18] is the Object Management Group's (OMG) standard. SPEM 2.0 provides the necessary concepts for modelling, documenting, interchanging, and presenting systems and software development processes. The conceptual framework of SPEM 2.0 consists of *Method Content* and the *Process*. *Method Content* allows users to define reusable process content, i.e., partially ordered *tasks*, *work products* (which can be a type of artifact, deliverable, or outcome), *roles* and *guidances*, and the *Category* such as *disciplines*, *role sets*, *domains* and *tools*. *Process* describes the systematic development processes as sequences of *phases* and *milestones* for the specific types of projects. To define a *process*, tasks can be grouped to form an *activity* and a set of nested activities can be grouped into *iteration* (to indicate that the set can be repeated more than once). A process can be a *capability pattern*, which describes reusable clusters of activities or a *delivery process*, which describes a complete end-to-end project life-cycle. Table 1 shows the main structural elements for defining the process in SPEM 2.0.

Table 1: Process modelling elements in SPEM 2.0

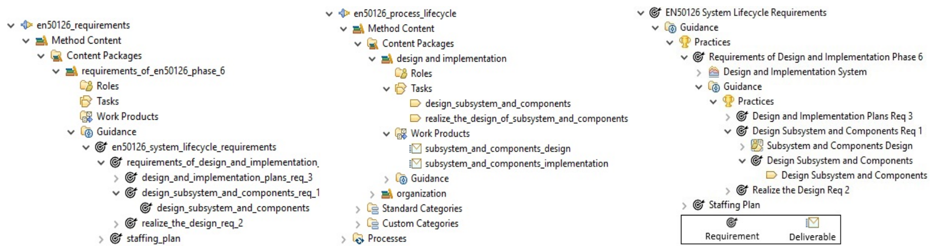


Delivery Process	Capability Pattern	Activity	Iteration	Phase	Milestone	Task
Role	**Work Products**	**Guidance**	**Practice**	**Role Sets**	**Tool**	**Disciplines**

SPEM 2.0 supports variability management in the *Method Content* package, which allows elements to modify or reuse elements in other content packages without directly modifying the original content. SPEM 2.0 defines five types of variability relationships: *not assigned* (na)—the default value, *contributes*, *replaces*, *extends*, and *extends and replaces* [18]. In the scope of this paper, we consider *Extends* variability and *Contributes* variability.

2.3 Modelling Standards and Safety Plans in EPF Composer

EPF Composer is an extensible process framework, based on the Unified Method Architecture (UMA) metamodel, which covers most of the SPEM 2.0 [18] concepts, needed for our purposes. It is worth to highlight that EPF Composer has been recently ported from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 in the context of the AMASS project [13]. As presented in [6, 16] and [3], based on Mc Isaac's approach [14] conceived for the commercial version of EPF Composer, EPF Composer can be used to model standards and safety plans, as well as to show that plans comply with standards. In EPF Composer, *method plugins* are containers of process related information (i.e., Method Content and Processes), while a *configuration* is a selection of sub-sets of library content to be shown in the browsing perspective. To model the requirements listed in the standards, the guidance type *Practice* can be customized with an icon in a separate plugin *(customized_icon)*. The *standard requirements plugin* captures the standard's requirements and has the variability relationship *Extends* with the previously mentioned *customized_icon* plugin. Requirements can be nested (i.e., a requirement inside another requirement to respect the nesting existing in the standards), as shown in Figure 1a. The *process lifecycle plugin* defines the process life-cycle (i.e., content elements, categories and processes), as shown in Figure 1b. To define the mapping, standard requirements are copied in *mapping requirements plugin*. These copied requirements have a variability relationship *Contributes* with original requirements modelled in *standard requirements plugin*. In addition, the links between process elements (such as tasks) to each "standard requirement" have been established through "references" tab. The mapped requirements can be grouped in *Custom Categories* to facilitate their visualization in the browsing perspective. To do so, a Custom Category, named *Mapped Requirements*, is created in the *mapping requirements plugin* and all requirements are assigned through the "Assign" tab. Figure 1c shows the mapped requirements in EPF Composer.



```
en50126_requirements
  Method Content
    Content Packages
      requirements_of_en50126_phase_6
        Roles
        Tasks
        Work Products
        Guidance
          en50126_system_lifecycle_requirements
            requirements_of_design_and_implementation_
              design_and_implementation_plans_req_3
              design_subsystem_and_components_req_1
                design_subsystem_and_components
              realize_the_design_req_2
            staffing_plan
```

(a)

```
en50126_process_lifecycle
  Method Content
    Content Packages
      design and implementation
        Roles
        Tasks
          design_subsystem_and_components
          realize_the_design_of_subsystem_and_components
        Work Products
          subsystem_and_components_design
          subsystem_and_components_implementation
        Guidance
      organization
    Standard Categories
    Custom Categories
  Processes
```

(b)

```
EN50126 System Lifecycle Requirements
  Guidance
    Practices
      Requirements of Design and Implementation Phase 6
        Design and Implementation System
        Guidance
          Practices
            Design and Implementation Plans Req 3
            Design Subsystem and Components Req 1
              Subsystem and Components Design
              Design Subsystem and Components
                Design Subsystem and Components
            Realize the Design Req 2
      Staffing Plan
```

(c)

Fig. 1: A cut of the EN 50126-compliant models in EPF Composer -focus on Phase 6

2.4 CACM and OpenCert within the AMASS platform

Common Assurance and Certification Metamodel (CACM) is created for the AMASS platform. CACM consists of several packages/metamodels. More specifically, CACM incorporates: 1) the *System Component Metamodel*, which is based on the modelling language (CHESSML) [7], to support the specification of system-specific details and decisions; 2) the *Assurance Case Metamodel*, which is based on the Structured Assurance Case Metamodel (SACM) [17], to support the modelling assurance cases; 3) the *Compliance Management Metamodel* group, which, in turn, consists of several metamodels such as *Process Definition Metamodel*—based on the UMA metamodel, *Assurance Project Definition*, *Baseline Definition Metamodel* and etc. These metamodels focus on what is planned to be done in a project. The *Evidence Management Metamodels* group based on three OpenCert metamodels: *Artifact Metamodel*, *Executed Process Metamodel* and *Traceability Metamodel* (AssuranceAsset). These metamodels deal with what has actually been done.

Three tools compose the AMASS platform: EPF Composer, OpenCert[6] and CHESS Toolset[7]. The interested reader may refer the AMASS platform presentation, hosted within the new OpenCert space[8].

In this paper, we focus on *Baseline Definition Metamodel*, *Artifact Metamodel*, *Executed Process Metamodel* of OpenCert tool. The main elements of metamodels and their semantics are given in the following subsections.

Baseline Definition Metamodel The *Baseline Definition Metamodel* (BDM) defines what is planned to be complied with a concrete standard, in a specific assurance project. In the following list, we recall the BDM elements used in the remaining of this paper:

- *BaseFramework* is a main container to model the concepts against which safety and system engineering aspects of a given system are developed and assessed.
- *BaseActivity* is the first-class modelling entity of process specifications, which describes a phase, activity or tasks depending on the activity granularity level defined in a standard or company process. The base activity can be decomposed into one or more fine-grained base activities, called *subActivity*.
- *BaseRequirement* specifies the criteria (e.g., objectives) that a base framework defines (or prescribes) to comply with it.

Artefact Metamodel The *Artefact Metamodel* specifies the classes and relationships that can be used to support the reasoning of managed artefacts as an evidence of standards compliance. In the following list, we recall the elements of *Artefact Metamodel* used in the remaining of this paper:

- *ArtefactModel* defines the root element of a model representing a set of Artefacts.
- *ArtefactDefinition* specifies a distinguishable abstract unit of data to manage in an assurance project, which represents the whole life-cycle resulting from the evolution, in different versions of Artefacts. In particular, it is a template of a work product involved in an activity.
- *Artefact* describes the instance of artefacts characterised for a version and a set of resources modelling tangible artefact resources or files. An Artefact can be composed of other artefacts or artefact parts.

Executed Process Metamodel The *Executed Process Metamodel* supports the specification of process-specific compliance needs that might have to be considered in an assurance project, such needs include not only the activities to execute, but also artefacts to manage. In the following list, we recall the elements of *Executed Process Metamodel* used in the remaining of this paper:

[6] `https://www.polarsys.org/projects/polarsys.opencert`
[7] `https://www.polarsys.org/chess/index.html`
[8] `https://www.polarsys.org/opencert/`

- *ProcessModel* is a container of root elements to model a set of Process elements. The Process model corresponds to the actual execution of a process with data related to the results to the process.
- *Activity* models a unit of work performed in a product life-cycle. An Activity is a specification of an activity already executed.
- *Person* models individuals that are involved in a product life-cycle.
- *Tool* models software tools used in a product life-cycle.
- *Organization* corresponds to the groups of people (e.g., companies, societies, associations, etc.) that are involved in a product life-cycle.
- *Technique* is used in the Activity to generate the produced Artefacts.

2.5 Model-driven Engineering

As summarised in [11], Model-driven Engineering (MDE) is a model-centric software development methodology aimed at raising the software at different levels of abstraction and increasing automation in software development. For automation purposes, model-to-model transformation is used to refine models. In particular, model-to-model transformation transforms the source model (compliant with one metamodel) into a target model compliant with the same or a different metamodel. A standard transformation can be defined as a set of rules to map source to the target. A transformation can be defined by using transformation languages. Epsilon Transformation Language (ETL)[9] is a hybrid, rule-based model-to-model transformation language and provides the enhanced flexibility to transform arbitrary number of source models to an arbitrary number of target models. An ETL transformation is typically organised in modules *ETLModule* and each module can contain any number of transformation rules *TransformationRule* and *Epsilon Object Language (EOL) operations*.

3 Tool-support Model-based Method

In this section, we present our tool-supported model-based method for facilitating the safety approval process. The overview, given in SPEM2.0, of our method is illustrated in Figure 2. As the activity diagram illustrates, for getting the approval of the safety plans, the compliance between the safety plans and the CENELEC series requirements has to be shown. All this is done in EPF Composer by modelling the requirements, the plans, and the compliance (shown, in this paper, via a simple mapping between standards requirements and safety plans through references in EPF Composer as shown in Section 2.3, see Figure 1. Alternatively, compliance could be explained via argumentation as presented in [16], where process-based arguments (model and diagram) can be derived automatically from process models. Next, the compliant evidence is given to the certification body for approval, afterwards OpenCert tool is used for the execution of the process (safety plan).

For facilitating the compliance between the executed process (including the corresponding evidence) and the planned process, the transformations of standards requirements and planned process from EPF Composer into baselines, post-planning process

[9] See `https://www.eclipse.org/epsilon/doc/etl/`

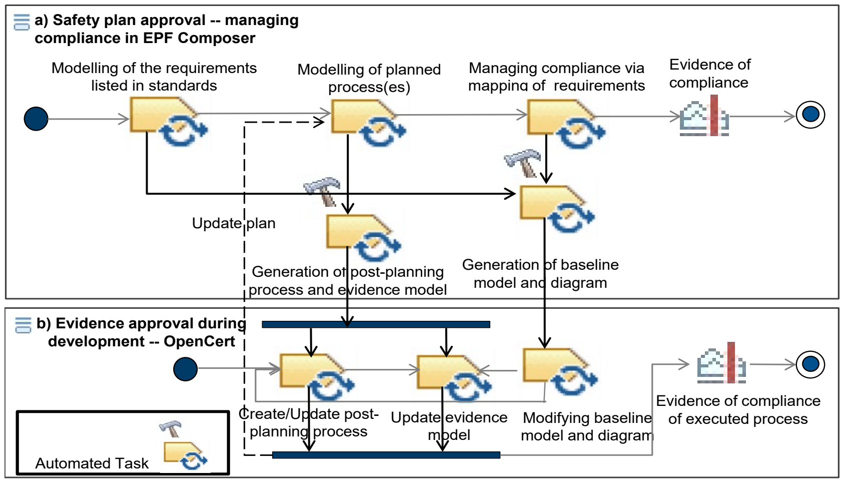

a) Safety plan approval -- managing compliance in EPF Composer
Modelling of the requirements listed in standards
Modelling of planned process(es)
Managing compliance via mapping of requirements
Evidence of compliance
Update plan
Generation of post-planning process and evidence model
Generation of baseline model and diagram
b) Evidence approval during development -- OpenCert
Create/Update post-planning process
Update evidence model
Modifying baseline model and diagram
Evidence of compliance of executed process
Automated Task

Fig. 2: Overview of the proposed method for facilitating the safety approval process

and evidence models into OpenCert are performed. In particular, the requirements modelled in EPF Composer as “Practice" under the Content Packages are proceeded to generate the baseline model and diagram using implemented *Baseline Generator plugin* (see Section 3.1); while the delivery process modelled in EPF Composer is proceeded to generate an evidence model and a process model in OpenCert by using *Process and Evidence Models Generator plugin* (see Section 3.2). These transformations help process engineers to get the baseline model and diagram, a first version of their post-planning process model and evidence model, which enables the evolution and traceability of models during the development phase. The baseline model and diagram, post-planning process and evidence models are generated locally as well as in the CDO[10] (Connected Data Objects) repository. CDO is a development-time model repository as well as a run-time persistence framework, which offers transactions with save points, change notifications, queries, transparent temporality, and etc. OpenCert supports engineers to update or evolve the models during the development phase (Figure 2b) and evidence of compliance is provided for review.

3.1 Generating Baseline Model from Standard Requirements

In this subsection, we explain how we generate the baseline model and diagram from standard requirements for providing the convincing justification to the certification body about compliance means. For this, the mapping between standard requirements compliant with SPEM/UMA and baseline elements compliant with BDM (part of CACM) has been implemented. In order to get the nested requirements and differentiate between them (for example, which process element (i.e., phase, activity, task) is mapped to a standard requirement), we retrieve the information from the *mapping requirements*

[10] `http://www.eclipse.org/cdo/`

plugin through “activityReferences” and “contentReferences”. The main mapping between these metamodels is described in Table 2. In particular, the *ContentPackage* that contains the requirements is mapped into a *BaseFramework*, whereas the top-level requirement *Practice* related to the Delivery Process or Capability pattern is mapped to the *BaseActivity*. These requirements are decomposed into sub-requirements associated to phases, in turn, for each phase all sub-requirements associated to activities and so on; until the sub-requirements associated to tasks are reached we mapped them into *BaseRequirements* in the Baseline model. Id, name and description of requirements are mapped into Id, name and description of baseline elements.

Table 2: Mappings Concepts of Requirements

SPEM/UMA	**BDM**
ContentPackage	BaseFramework
Practice (top-level requirement)	BaseActivity
subPractice (requirement associated to Phase/Activity)	subActivity
subPractice (requirement associated to Task)	BaseRequirement
Id, name and description	Id, name and description

Algorithm 1 Generating Baseline Model

```
Input: UMA: ContentPackage, ProcessComponent, Baseline Definition Metamodel, Executed Process Metamodel
Output: BaseFramework
while $childPackages.isTypeOf.ContentPackage = “CoreContent”$ do
  $BaseFramework \leftarrow getElementsByTagName(uma : ContentPackage)$
  Transform
  $(BaseFramework \leftarrow ContentPackage)$;
  // Map all three attributes for all elements <element>.id,
   <element>.name, <element>.briefDescription
  for $contentElements.isTypeOf(uma : Practice)$ do
    for $activityReferences$ in $Practice.activityReferences$ do
      $(BaseActivity \leftarrow Practice)$;
      for $subPractice$ in $Practice.subPractices()$ do
        for $activityReferences$ in $subPractices.activityReferences$ do
          $(subActivity \leftarrow subPractice)$;
        end for
        // subPractices linked with role, task, work product
        for $contentReferences$ in $subPractices.contentReferences$ do
          $(BaseRequirement \leftarrow subPractice)$;
        end for
      end for
    end for
  end for
end while
```

The mapping is achieved by using ETL, in particular, a *Baseline Generator plugin* has been implemented in the AMASS platform, which automatically transforms the requirements into baseline model and diagram. The generated baseline model and diagram are visualized via the Baseline editor in OpenCert. The generated baseline model and diagram are also stored in the CDO Repository. Algorithm 1 shows the skeleton of generation of baseline model and diagram.

3.2 Generating Post-planning Processes and Evidence Models

In this subsection, we present our algorithmic solution for the generation of the post-planning process and the evidence model in OpenCert, from the planned process, modelled in EPF Composer. The mapping is focused on the *Work Breakdown Structure* of Delivery Process in EPF Composer. In particular, a Delivery Process in EPF Composer is contained in the metamodel class *ProcessComponent* which provides additional information to the process description like its version, authors or team profiles required for the execution of the process. However, the user does not explicitly require creating a ProcessComponent in the EPF Composer; they are automatically created each time a delivery process or capability pattern is created. The main mappings between UMA/SPEM and CACM Executed Process and Artefact metamodels are described in Table 3. OpenCert provides the Assurance Process and Evidence Model wizard to visualise generated process model and evidence model, respectively.

Table 3: Mappings Concepts of Process and Artefacts

SPEM/UMA	**Executed Process and Aretfact Metamodels**
ProcessComponent	ProcessModel, ArtefactModel
CapabilityPattern	Activity
Activity, Phase, Iteration, TaskDescriptor, Milestone	subActivity
RoleDescriptor	Person
Guideline, Practice	Technique
ToolMentor	Tool
RoleSet, TeamProfile	Organization
WorkProductDescriptor	ArtefactDefinition, Artefact
Id, name and description	Id, name and description

In general, evidences are specified and managed by evidence models. Within this model, objects for Artefacts and Artefact Models can be created. The semantics of *ArtefactDefinition* and *Artefact* are slightly different in CACM and UMA metamodels. In the case of CACM, *ArtefactDefinition* is a template of a work product involved in an activity, whereas an *Artefact* represents the specific work product involved in the activity which uses particular template. On the other hand, in UMA, *Artifact* is an element that belongs to the Method Content package and *WorkProductDescriptor* is an instantiation of an artefact in the context of an activity. Therefore, an ArtefactDefinition and an initial version of Artefact are generated from the WorkProductDescriptor, shown in

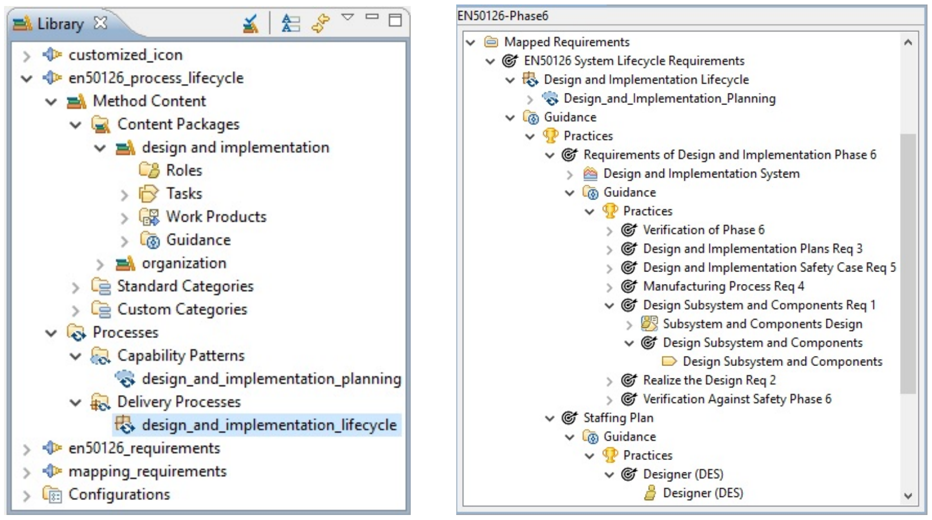
Algorithm 2 Generating Post-planning Process and Evidence Model

```
Input: UMA: ProcessComponent, Executed Process Metamodel, Aretfact Metamaodel
Output: ProcessModel, ArtefactModel
while $ProcessComponent.isTypeOf(DeliveryProcess)$ do
  $ProcessModel \leftarrow getElementsByTagName(uma : ProcessComponent)$
  Transform
  $(ProcessModel\ \&\ ArtefactModel \leftarrow ProcessComponent)$;
  for all CapabilityPattern do
    $(Activity \leftarrow CapabilityPattern)$;
    // Map all three attributes for all elements <element>.id,
     <element>.name, <element>.briefDescription
    if $CapabilityPattern.breakdownElements.isTypeOf(Phase)! = null$ then
      for $Phase$ in $CapabilityPattern.breakdownElements.isTypeOf(Phase)$ do
        $(Activity \leftarrow Phase)$;
      end for
    end if
    if $Activity.getTaskDescriptors(Activity.breakdownElements)! = null$ then
      for $TaskDescriptors$ in $Activity.getTaskDescriptors(Activity.breakdownElements)! =$
      $null$ do
        $(Activity \leftarrow TaskDescriptors)$;
        if $TaskDescriptor.WorkProductDescriptor! = null$ then
          for $WorkProductDescriptor$ in $TaskDescriptor.WorkProductDescriptor$ do
            Call operations getexternalInput(); getoptionalInput(); getmandatoryInput();
            getoutput();
            $(ArtefactDefinition\ \&\ Artefact \leftarrow WorkProductDescriptor)$;
          end for
        end if
      end for
    end if
  end for
end while
```

Table 3. The mapping is achieved by using ETL, in particular, *Process and Evidence Generator* plugin has been implemented in the AMASS platform. Algorithm starts by searching the ProcessComponent if it is the type of Delivery Process and considers the *Work Breakdown Structure* (decomposed) linked elements such as phases, activities etc. Algorithm 2 shows the skeleton of our transformation.

4 An Illustrative Example

In this section, we apply our tool-supported method to show how it facilitates the safety plan acceptance and approval process defined in the EN 50129 standard. Our focus is on the safety demonstration for a generic product (i.e. independent of application) as indicated in EN 50129, Part 5.5.2. Moreover, our focus is limited to Phase 6 (Design and Implementation) of the EN 50126-RAMS life-cycle, which must be taken into consideration for the definition of the portion of the safety plan regarding design and implementation. Based on that, we model: the custom practice for representing a generic requirement modelling element, the requirements from that phase (which inherit from the generic requirement), the portion of the safety plan, which is expected to comply with those requirements, and the compliance (achieved via a mapping through references in *mapping requirements plugin*). The basic compliance between the requirements listed in the Phase 6 (Design and Implementation) of EN 50126 standard and planned process is shown in Figure 3b.


Library
customized_icon
en50126_process_lifecycle
Method Content
Content Packages
design and implementation
Roles
Tasks
Work Products
Guidance
organization
Standard Categories
Custom Categories
Processes
Capability Patterns
design_and_implementation_planning
Delivery Processes
design_and_implementation_lifecycle
en50126_requirements
mapping_requirements
Configurations
EN50126-Phase6
Mapped Requirements
EN50126 System Lifecycle Requirements
Design and Implementation Lifecycle
Design_and_Implementation_Planning
Guidance
Practices
Requirements of Design and Implementation Phase 6
Design and Implementation System
Guidance
Practices
Verification of Phase 6
Design and Implementation Plans Req 3
Design and Implementation Safety Case Req 5
Manufacturing Process Req 4
Design Subsystem and Components Req 1
Subsystem and Components Design
Design Subsystem and Components
Design Subsystem and Components
Realize the Design Req 2
Verification Against Safety Phase 6
Staffing Plan
Guidance
Practices
Designer (DES)
Designer (DES)

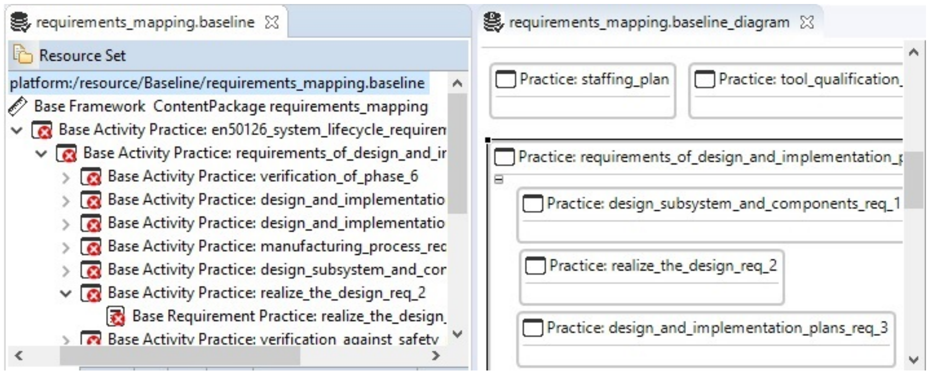
(a) (b)

Fig. 3: Mapping of planned process and standards in EPF Composer

This modelling activity is performed in EPF Composer by following the guidelines mentioned in Section 2.3. As a results, as shown in Figure 3a, four EPF Composer plugins are created. Concerning the plan, we model: activities, work products, methods, roles, etc. Concerning roles, since EN 50126 is less prescriptive than EN 50128 [8], we have decided to borrow from EN 50128 to plan the responsibilities and competence required at system design level. Due to space limits, the complete visualization of the result of our modelling activity cannot be shown. The interested reader can access the complete EPF Composer project regarding Phase 6 [15].

The OpenCert tool allows users to model baselines as requirements. Instead of manually creating the baselines, the baseline model and diagram are automatically generated from the ContentPackage using our *Baseline Generator plugin* (see Section 3.1). Figure 4 shows generated baseline model and diagram, compliant to the BDM that are visualised in baseline editor in OpenCert. The generated baseline model and diagram are also stored in the CDO Repository.

Finally, for the getting the approval for the evidence produced during the development, compatibility between the executed process and the planned process has also to

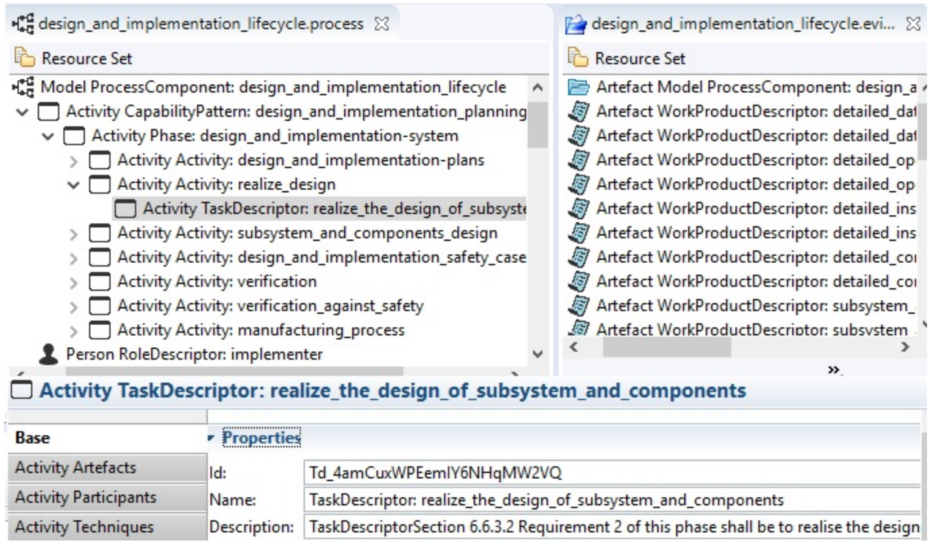

requirements_mapping.baseline

Resource Set
platform:/resource/Baseline/requirements_mapping.baseline
Base Framework ContentPackage requirements_mapping
Base Activity Practice: en50126_system_lifecycle_requiren
Base Activity Practice: requirements_of_design_and_ir
Base Activity Practice: verification_of_phase_6
Base Activity Practice: design_and_implementatio
Base Activity Practice: design_and_implementatio
Base Activity Practice: manufacturing_process_rec
Base Activity Practice: design_subsystem_and_cor
Base Activity Practice: realize_the_design_req_2
Base Requirement Practice: realize_the_design_
Base Activity Practice: verification against safety

requirements_mapping.baseline_diagram

Practice: staffing_plan
Practice: tool_qualification_
Practice: requirements_of_design_and_implementation_
Practice: design_subsystem_and_components_req_1
Practice: realize_the_design_req_2
Practice: design_and_implementation_plans_req_3

Fig. 4: Generated baseline model and diagram


design_and_implementation_lifecycle.process

Resource Set
Model ProcessComponent: design_and_implementation_lifecycle
Activity CapabilityPattern: design_and_implementation_planning
Activity Phase: design_and_implementation-system
Activity Activity: design_and_implementation-plans
Activity Activity: realize_design
Activity TaskDescriptor: realize_the_design_of_subsyste
Activity Activity: subsystem_and_components_design
Activity Activity: design_and_implementation_safety_case
Activity Activity: verification
Activity Activity: verification_against_safety
Activity Activity: manufacturing_process
Person RoleDescriptor: implementer

design_and_implementation_lifecycle.evi...

Resource Set
Artefact Model ProcessComponent: design_a
Artefact WorkProductDescriptor: detailed_dat
Artefact WorkProductDescriptor: detailed_dat
Artefact WorkProductDescriptor: detailed_op
Artefact WorkProductDescriptor: detailed_op
Artefact WorkProductDescriptor: detailed_ins
Artefact WorkProductDescriptor: detailed_ins
Artefact WorkProductDescriptor: detailed_coi
Artefact WorkProductDescriptor: detailed_coi
Artefact WorkProductDescriptor: subsystem_
Artefact WorkProductDescriptor: subsystem

Activity TaskDescriptor: realize_the_design_of_subsystem_and_components

Base
Activity Artefacts
Activity Participants
Activity Techniques

Properties

Id:	Td_4amCuxWPEemlY6NHqMW2VQ
Name:	TaskDescriptor: realize_the_design_of_subsystem_and_components
Description:	TaskDescriptorSection 6.6.3.2 Requirement 2 of this phase shall be to realise the design

Fig. 5: Generated post-planning process and evidence model

be shown or the deviations should be tracked and explained. For this, the safety plans compliant with the EN 50126 requirements (including the work products) modelled in EPF Composer are transformed into post-planning process and evidence model in OpenCert. Specifically, the transformation is performed using the *Process and Evidence Models Generator* plugin by right-clicking on ProcessComponent (i.e. delivery process) from the EPF Composer (see Section 3.2). The generated evidence model listing all the artefacts to be produced during the execution phase. The generated post-planning process and evidence model can evolve during the life-cycle. Figure 5 shows generated post-planning process and evidence model in OpenCert. The generated models are also stored in the corresponding in the CDO Repository under “PROCESSES” and “EVIDENCE” folder. Both transformations took few seconds to generate the baseline (model and diagram), post-planning process and evidence model. Without the automatic generation of these models, engineers would have to model them manually from scratch which requires huge effort, also managing manual traceability between executed process and the planned process is very difficult.

5 Related Work

In the literature, as far as we know, no work has addressed the facilitation of the safety plan acceptance and approval process (/certification liaison process) during both phases: planning and execution. However, in the literature, several works exist on execution of process models and on transformations from models created in technological spaces for process definition to models derived within technological spaces for their execution/exchange (Gallina et al. [12] propose an extension of SPEM 2.0, called S-TunExSPEM, for modelling and exchanging safety processes; Bendraou et al. [5] present a model-driven approach, which includes the mapping between UML4SPM, used for the definition of software processes, and WS-BPEL, used for process execution; Alajrami et al. [2] propose and extension of SPEM2.0, called EXE-SPEM for enabling process models execution on the Cloud). In addition, Adedjouma et al. [1] present an approach that transforms the text-based standards to a tree-like structure relying upon the JSON transducer, and then from the JSON tree-like structure to a graphical BPMN model for easy visualisation and navigation. In Adedjouma et al.'s approach non-textual standard elements such as figures, tables are formatted manually by the user. Our work does not automate the digitalisation of the standards yet. However, it provides full guidance for their manual digitalisation, including complex recommendation tables, which populate the standards. Schoitsch et al. [20] propose the certification process in DECOS (Dependable Embedded Components and Systems), which is implemented in a modular way and uses the concept of generic safety cases. The proposed approach is supported by the Generic Test Bench to generate the safety cases by providing generic v-plans for safety standards, documentation support in order to generate the validation report from the completed v-plans and built-in user guidance in terms of a help file. As compared to these works, our tool-supported method facilitates the safety approval process by offering a browsing perspective of compliance management, within EPF Composer during the planning phase and within OpenCert via the evidence management during the execution phase. Our method supports the modelling of plans in compliance with the standards and the automatic generation of post-planning process (including corresponding evidence) required for the execution phase.

6 Conclusion and Future Work

EN 50129-compliant safety plan acceptance and approval process (similar to the DO-178C-compliant certification liaison process process) requires the interaction between the applicant and the certification body in order to get approval first for the plans and then for the evidence (which represents the substantiation of the plans). This process is delicate and time consuming due to the necessity of showing that all pieces of evidence produced comply with the CENELEC standard series. In this paper, we have presented a tool-supported method for facilitating such process. Specifically, our method supports: the modelling of the standards, the modelling of the plans in compliance with the standards, the automatic generation of corresponding process-related representations needed for the execution phase and for impact-change tracking. As a consequence, it facilitates the compliance demonstration during the planning phase and the manual review of the safety plan after the execution of each safety life-cycle phase to track alterations or extensions. We have illustrated the usage of our method for facilitating the approval of a portion of an EN 50126/9-compliant safety plan targeting the design specification.

At the current stage of development, our method automatically generates the process-related representations needed for the execution phase. However, in case of alterations and/or extensions made during the execution phase, back transformation from the executed processes to the planned processes is not supported yet. As future work, we intend to investigate the back-propagation of the changes. Based on our gathered experience, such propagation could be achieved by defining similar transformation rules, as presented in this paper, but on the opposite directions. We also intend to conduct a proper evaluation of the approach to achieve a quantitative measurement of the gain that users might get via application of our method. To do that, we will not only consider a generic safety plan, but we will consider its instantiation at a specific project level for the design and implementation of a real subsystem. This in-depth evaluation is planned to be carried out in the context of the AMASS case study 6 (Automatic Train Control Formal Verification) in cooperation with Alstom.

Acknowledgment

This work is supported by EU and VINNOVA via the ECSEL Joint Undertaking under grant agreement No. 692474, AMASS project. We thank Inmaculada Ayala for her contribution on requirements modelling using customised elements in EPF Composer.

References

1. Adedjouma, M., Pedroza, G., Smaoui, A., Dang, T.K.: Facilitating the adoption of standards through model-based representation. In: Proceedings of the 23rd International Conference on Engineering of Complex Computer Systems (ICECCS '18), Melbourne, Australia, December 12-14, 2018. (2018)
2. Alajrami, S., Gallina, B., Romanovsky, A.: Exe-spem: Towards cloud-based executable software process models. In: 4th International Conference on Model-Driven Engineering and Software Development, MODELSWARD. pp. 517–527 (2016)
3. AMASS: AMASS User guidance and Methodological framework. `https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/D2.5_User-guidance-and-methodological-framework_AMASS_Final.pdf` (2018), (Last accessed: March 5, 2019)
4. AMASS: AMASS platform validation D2.9. `https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D2.9_AMASS-platform-validation_AMASS_Final.pdf` (2019), (Last accessed: March 5, 2019)

5. Bendraou, R., Jezéquél, J.M., Fleurey, F.: Combining aspect and model-driven engineering approaches for software process modeling and execution. In: Trustworthy Software Development Processes. pp. 148–160. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
6. Castellanos Ardila, J.P., Gallina, B., Ul Muram, F.: Enabling Compliance Checking against Safety Standards from SPEM 2.0 Process Models. In: 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, Prague, Czech Republic, August 29-31, 2018. pp. 45–49 (2018). https://doi.org/10.1109/SEAA.2018.00017
7. CHESS-Team: CHESSML https://www.polarsys.org/chess/start.html (2018)
8. European Commitee for Electrotechnical Standardization (CENELEC): EN 50128 - Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems (2011)
9. European Commitee for Electrotechnical Standardization (CENELEC): EN 50126-1: Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS), Part 1 Generic RAMS process (2017)
10. European Commitee for Electrotechnical Standardization (CENELEC): EN50129: Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling (2018)
11. Gallina, B.: A Model-driven Safety Certification Method for Process Compliance. In: 2nd International Workshop on Assurance Cases for Software-intensive Systems, joint event of ISSRE, Naples, Italy, November 3-6, 2014. pp. 204–209. IEEE (2014). https://doi.org/10.1109/ISSREW.2014.30
12. Gallina, B., Pitchai, K.R., Lundqvist, K.: S-tunexspem: Towards an extension of SPEM 2.0 to model and exchange tunable safety-oriented processes. In: Software Engineering Research, Management and Applications [selected papers from the 11th International Conference on Software Engineering Research, Management and Applications, SERA 2013, Prague, Czech Republic, August 7-9, 2013]. pp. 215–230 (2013). https://doi.org/10.1007/978-3-319-00948-3_14
13. Javed, M.A., Gallina, B.: Get EPF Composer back to the future: A trip from Galileo to Photon after 11 years. EclipseCon, Toulouse, France, June 13-14,. `http://www.es.mdh.se/publications/5091-Get_EPF_Composer_back_to_the_future__A_trip_from_Galileo_to_Photon_after_11_years` (2018)
14. McIsaac, B.: IBM Rational Method Composer: Standards Mapping. Tech. rep., IBM Developer Works (2015)
15. Muram, F.U., Gallina, B.: EPF Composer Library for EN 50126-9 compliant process authoring, limited to Phase 6. `https://www.dropbox.com/sh/1o7cf12nqvmyvqc/AACi0EZymqzbQJKinutcNAzsa?dl=0` (2019), (Last accessed: March 5, 2019)
16. Muram, F.U., Gallina, B., Rodriguez, L.G.: Preventing Omission of Key Evidence Fallacy in Process-based Argumentations. In: 11th International Conference on the Quality of Information and Communications Technology (QUATIC), Coimbra, Portugal, September 4-7, 2018. pp. 65–73. IEEE (2018). https://doi.org/10.1109/QUATIC.2018.00019
17. Object Management Group (OMG): Structured Assurance Case Metamodel (SACM), Version 2.0. `https://www.omg.org/spec/SACM/2.0` (2018), (Last accessed: March 5, 2019)
18. OMG: Software & Systems Process Engineering Metamodel Specification (SPEM), Version 2.0. `http://www.omg.org/spec/SPEM/2.0/` (2008), (Last accessed: March 5, 2019)
19. RTCA Inc: Software Considerations in Airborne Systems and Equipment Certification, RTCA DO-178C (EUROCAE ED-12C). Washington DC (2011)
20. Schoitsch, E., Althammer, E., Sonneck, G., Eriksson, H., Vinter, J.: Modular certification support - the decos concept of generic safety cases. In: 2008 6th IEEE International Conference on Industrial Informatics. pp. 258–263 (July 2008). https://doi.org/10.1109/INDIN.2008.4618105