S-TunExSPEM: Towards an Extension of SPEM 2.0 to Model and Exchange Tunable Safety-oriented Processes

Barbara Gallina, Karthik Raja Pitchai and Kristina Lundqvist

Abstract Prescriptive process-based safety standards (e.g. EN 50128, DO-178B, etc.) incorporate best practices to be adopted to develop safety-critical systems or software. In some domains, compliance with the standards is required to get the certificate from the certification authorities. Thus, a well-defined interpretation of the processes to be adopted is essential for certification purposes. Currently, no satisfying means allows process engineers and safety managers to model and exchange safety-oriented processes. To overcome this limitation, this paper proposes S-TunExSPEM, an extension of Software & Systems Process Engineering Meta-Model 2.0 (SPEM 2.0) to allow users to specify safety-oriented processes for the development of safety-critical systems in the context of safety standards according to the required safety level. Moreover, to enable exchange for simulation, monitoring, execution purposes, S-TunExSPEM concepts are mapped onto XML Process Definition Language 2.2 (XPDL 2.2) concepts. Finally, a case-study from the avionics domain illustrates the usage and effectiveness of the proposed extension.

Key words: DO-178B, safety-oriented processes, process modelling, SPEM 2.0, process exchange, XPDL 2.2, process reuse.

1 Introduction

The increasing awareness of software development being a complex task has since the 1980's received increased attention from the research community working on

Barbara Gallina
Mälardalen University, P.O. Box 883, SE-72123 Västerås, Sweden. e-mail: barbara.gallina@mdh.se

Karthik Raja Pitchai
Mälardalen University, P.O. Box 883, SE-72123 Västerås, Sweden. e-mail: kpi10001@student.mdh.se

Kristina Lundqvist
Mälardalen University, P.O. Box 883, SE-72123 Västerås, Sweden. e-mail: kristina.lundqvist@mdh.se

engineering software processes [11]. Software processes can be defined as coherent sets of policies, organizational structures, technologies, procedures, and artefacts that are needed to conceive, develop, deploy, and maintain a software product [11]. The research motivation surrounding software processes is based on the assumption that there is a correlation between the quality of the process and the quality of the software developed. According to what is stated in [15], each life-cycle phase may represent a source of faults that if not handled lead to system failures causing serious incidents. To avoid such failures, processes must be enhanced by preventing or removing potential faults. To enhance processes, in the context of safety standards, best systems and software development practices have been collected and prescriptive processes have been defined. More specifically, these processes mandate the activities to be executed, the work-products to be provided, the qualifications needed to execute the activities, the tools to be used to execute the activities, and the guidelines to be followed. DO-178B [22], for instance, is the de facto standard for software development in civilian aircraft and its adoption is considered to be beneficial in contributing to the excellent record with remarkably few failures of avionics software [24]. Even though no strong correlation between the process and the product can be claimed in the context of dependable (safety-critical) systems, the enhancement of the processes permits the development of a deeper safety culture, leading the development team to act cautiously [16].

For certification purposes, in some domains, compliance with the processes defined within safety standards is mandatory. As investigated in [5], DO-178B leaves room for interpretation. In some cases, due to its lack of specificity in the guidelines, different users may come to different conclusions regarding the acceptability of a particular document or artefact based on their particular experience. Thus, as nicely put in [6] "for companies seeking a first-time certification, preparation for DO-178B can be a daunting challenge".

To ensure process understanding and thus eliminate inconsistencies in the process specification, a Process Modelling Language (PML) is necessary. Besides understanding, an adequate PML should permit users to document and exchange process models. In the literature, several PMLs are at disposal, e.g. Software & Systems Process Engineering Meta-Model 2.0 (SPEM 2.0). As recently reviewed in [23], SPEM 2.0 has obtained a significant acceptance and the research community is very active to propose extensions towards SPEM 3.0 in order to enhance its modelling capabilities, its executability, and its tool support. Thus, we decide to join this active research community to propose an extension, called S-TunExSPEM, to support the modelling as well as the exchange of safety-oriented processes. Our focus is limited to core process elements since our goal is to ease the adoption of S-TunExSPEM by providing an easy-to-digest PML. To define the set of core elements we have analyzed DO-178B to extract a list of key safety-related concepts. For these concepts we have defined: the abstract syntax by extending the SPEM 2.0 meta-model, the concrete syntax by providing new safety-oriented icons. Then, to enable process models interchange towards the usage of existing execution as well as monitoring and simulation engines, we have provided a mapping between S-TunExSPEM concepts and corresponding concepts of XML Process Definition Language 2.2 (XPDL 2.2). Finally, we have used S-TuneExSPEM to model processes for the development of avionics software.

The rest of the paper is organized as follows. In Sect. 2, we provide essential background information. In Sect. 3, we present S-TunExSPEM the proposed SPEM 2.0 extension that targets safety-oriented processes. In Sect. 4, we illustrate the usage and effectiveness of S-TunExSPEM by modelling a process taken from the avionics domain. In Sect. 5, we discuss related work. Finally, in Sect. 6 we present some concluding remarks and future work.

2 Background

In this section, we present the background information on which we base our work. In particular, in Sect. 2.1 we provide general information on safety-oriented processes and their role in the certification process. In Sect. 2.2 we provide essential information concerning the software development process defined in DO-178B. In Sect. 2.3, we briefly present SPEM 2.0, the process modelling language from which stems our extension. Finally, in Sect. 2.4, we briefly present XPDL 2.2, the process definition language onto which we map our SPEM 2.0 extension.

2.1 Safety-oriented processes and their role in certification

Prescriptive safety-oriented processes also known as safety life-cycles are systems (or software) development processes that prescribe best practices to be followed to achieve systems capable of managing safety by addressing the causes of accidents, namely hazards. Generally, a safety process requires safety analysts to identify and categorize the hazards according to domain-specific levels and risk assessment procedures. These levels, whose determination is not straightforward due to potential misconception/misuse/abuse [21], are called Design Assurance Levels (DALs) in the avionics domain in the context of DO-178B, Automotive Safety Integrity Levels (ASILs) in the automotive domain in the context of ISO 26262, and Safety Integrity Levels (SILs) in other domains that inherit the levels from IEC 61508. These levels (four or five depending on the specific standard) span from negligible to catastrophic hazards and they determine the number of objectives to be satisfied (eventually with independence) during the system (or software) development. Once hazards are classified, safety managers elicit safety requirements aimed at reducing risk. Then, they verify and validate the correct implementation and deployment of the elicited safety requirements throughout the safety life-cycle. It must be noted that it is not always possible to show that the systems developed meet the safety requirements.

Certification refers to the "process of assuring that a product or process has certain stated properties, which are then recorded in a certificate" [16]. Thus, for safety certification purposes, product and process-based arguments are needed to claim an acceptable level of safety. Process-based arguments are of particular value whenever confidence in product-based arguments is limited.

To provide convincing process-based arguments claiming for compliance, first of all it is necessary to achieve a well-defined and agreed-upon interpretation of the processes mandated within the standards [19]. Thus, adequate process modelling means are necessary and should be developed.

2.2 DO-178B

DO-178B [22] has been the de facto standard in the avionics domain. Currently, it is being replaced by a revised version (DO-178C), which addresses the inconsistencies of the previous document but preserves its basic and valuable principles. DO-178B provides guidance for the development of software for airborne systems and equipment. Its purpose is to guarantee a level of confidence in the correct functioning of the software developed in compliance with airworthiness requirements.

In this subsection, we provide a brief description of the software development process. This description is then used in Sect. 3 to extract process models. The software development process is constituted of four phases (requirements, design, coding and integration), which can be chained, if a waterfall process model is selected. The standard, however, does not impose a specific process model. In what follows, for each phase we provide its characteristics in terms of input/output, guidelines and roles. For sake of clarity, it must be noted that no role is explicitly assigned in DO-178B. Roles, however, can be inferred from the skills that are required and mentioned in the standard.

The **requirements phase** is characterized by:

Input: System requirements, hardware interface, system architecture, Software Development Plan, Software Requirements Standards.

Output: Software Requirements Data that include functional as well as non functional requirements.

Roles: requirement engineers in charge of functional requirements and quality (safety) experts in charge of non-functional requirements.

Guidelines: guidelines, defined in Section 5.1.2 of the standard, contain general as well as safety specific information.

The **design phase** is characterized by:

Input: Software development plan, Software Requirements Data, Software Design Standards.

Output: Design description.

Roles: designers in charge of the design decision related to functional requirements and quality (safety) experts in charge of the design decision related to non-functional requirements.

Guidelines: guidelines, defined in Section 5.2.2 of the standard, contain general as well as safety specific information.

The **coding phase** is characterized by:

Input: Software development plan, Design description, Software Code Standards.

Output: Source code and object code.

Roles: programmers in charge of the implementation decisions related to functional aspects of the design and quality (safety) experts in charge of the implementation decision related to non-functional aspects of the design.

Guidelines: guidelines, defined in Section 5.3.2 of the standard, contain general as well as safety specific information.

The **integration phase** is characterized by:

Input: Source code and object code, target computer, linking and loading data.

Output: Executable Object Code.

Roles: integration experts.

Guidelines: guidelines, defined in Section 5.4.2 of the standard, contain general as well as safety specific information.

With respect to the outputs that characterize the phases, a general remark is that for reuse purposes, outputs (e.g. Software Requirements Data) should be split to take into consideration the different views.

Details concerning how to break down the work within each phase are not provided in the standard. For sake of simplicity, we consider that each phase is constituted by a single task. Similarly, no specific tool is mentioned. However, at the organization-specific level, tools have to be planned (indeed a specific section called *Software development environment* is expected within the *Software Development Plan*) and used. The standard however recommends to guarantee traceability among the phases thus an additional task aimed at checking traceability can be considered.

2.3 SPEM 2.0

As recalled in the introduction, a software process can be defined as a coherent set of policies, organizational structures, technologies, procedures, and artefacts that are needed to conceive, develop, deploy, and maintain a software product. From this definition, it emerges that the core conceptual elements that are necessary to define a process are: guidelines, roles, tools, artefacts, and finally the breakdown structure to define the work to be executed.

In the literature, several PMLs that support those concepts are available [28, 1, 4]. SPEM 2.0 (Software & Systems Process Engineering Meta-Model 2.0) [18] is one of them and since it has appealing features in terms of standardization, reuse, tool-support, etc. (as surveyed in [4]) as well as in terms of an active community working towards its enhancement [23], it answers our expectations. SPEM 2.0 is the OMG's standard for systems and software process modelling and it is defined as a MOF-based meta-model. SPEM 2.0 meta-model is composed of seven main packages, which are briefly recalled in what follows.

The *Core* package defines concepts allowing for the foundation of the other packages. The *Method Content* package defines concepts allowing for the specification of a knowledge base of reusable process elements, as partially depicted in Fig. 1.

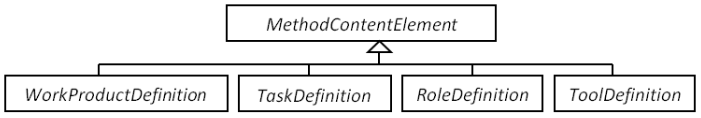


Fig. 1 Taxonomy of MethodContentElement.

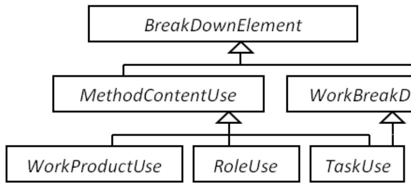


Fig. 2 Taxonomy of BreakDownElement.

The *Process Structure* package defines concepts allowing for the representation of process models composed of inter-related activities, roles (actual performers, called RoleUse), work-products (actual data, called WorkProductUse). The *Managed Content* package defines concepts such as Guidance allowing for the addition of descriptions in natural language to be attached to other process elements defined in other packages. The *Process with Method* package defines concepts such as Method Content Use elements for the integration of processes defined by using the concepts available in Process Structure with the instances of concepts available in Method Content. Fig. 2 depicts a sub-set of these concepts. The *Method Plugin* package defines mechanisms allowing for the reuse and management of method content and processes. The *Process Behaviour* package defines mechanisms and concepts (i.e. proxy meta-classes) allowing process elements to be linked to external models (e.g. UML 2.0 Activity Diagrams) for behavioural specification.

For a subset of the concepts that belong to the meta-model, graphical modelling elements (icons) are at disposal. In Table 1, we recall those elements for which we propose a safety-oriented decoration in Sect. 3.1. Tasks, roles and work-products (shortened as WP in Table 1) are commonly considered as process core elements [4]. Beside these elements, since we are focusing our work on safety-oriented processes, tools and guidances are also considered being core elements.

Table 1 Icons denoting Method Content (MC) and Method Content Use (MCU) elements

MC Elements					MCU Elements		
Task Definition	**Role Definition**	**Tool**	**WP Definition**	**Guidance**	**TaskUse**	**RoleUse**	**WPUse**
▭	👤	🔨	📄	📐	▭	👤	📄

2.4 XPDL 2.2

XML Process Definition Language 2.2 (XPDL 2.2) [27] is the current version of the XPDL specification recently issued by the Workflow Management Coalition (WfMC). XPDL 2.2 is a standard that defines an interchange format for process models. XPDL 2.2 syntax is specified by an XML schema. A process description in XPDL 2.2 is an XML document, which includes core modelling elements such as: Process, Activity, Transition, Participant, DataObject, and DataAssociation, and Application, Annotation. Below we recall the informal semantics of these elements and in Fig. 3 we provide the cut of XPDL 2.2 meta-model that includes them.

Annotation represents a piece of textual information that can be attached to activities or lanes. *Application* is used to specify the applications/tools invoked by the process. *Activity* represents a logical, self-contained unit of work. *Transition* represents the sequence-flow that relates two activities. Each individual transition has three elementary properties: the from-activity, the to-activity and the condition under which the transition is made. Activities and transitions are the key elements that form the *process*, which consists of an oriented graph composed of nodes (activities) and edges (transitions). *Participant* is used to specify the participants in the workflow, i.e., the entities that can execute work. There are six types of participant: ResourceSet, Resource, Role, OrganizationalUnit, Human, and System. *Pool* acts as the container for activities and transitions. *Lane* represents a performer information at the activity level. A lane is used to subdivide a pool and thus model who does what. *DataObject* (and related concepts such as DataInputs and DataOutputs) belongs to the set of new concepts, which have been introduced in XPDL 2.2. DataObject represents the primary construct for modelling data within a process and opens the possibility to model global as well as local variables and to model that data objects are transformed during the process execution [25]. DataInputs and DataOutputs are used to specify the I/O parameters needed by e.g. activities. *DataAssociation* represents a mapping between a data object on one end and a data input or data output on the other end. XPDL also offers extensibility mechanisms supported by the *extended attribute* modelling element. This element can be used to customize all the other XPDL 2.2 concepts.

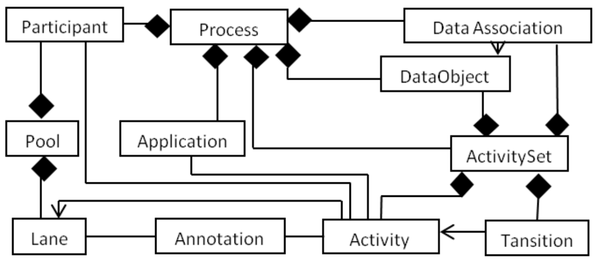


Fig. 3 Cut of XPDL 2.2 meta-model.

Currently, several commercial and open-source tools (e.g. process execution / monitoring / simulation engines) take XPDL descriptions in input and it is likely that soon new releases will be provided to support XPDL 2.2. This is why in Sect. 3 we propose a mapping onto XPDL 2.2 and not onto older versions. Moreover, we select XPDL 2.2 and not one specific execution language (e.g. Business Process Execution Language - BPEL) because by focusing on the exchangeability we can take advantage of the existing and various engines.

3 S-TunExSPEM

As discussed in the previous sections, development processes defined within safety standards exhibit safety-related concepts, which should be better supported by PMLs in order to allow process engineers and assessors to better communicate and easily identify process-based evidence. Thus, in this section we introduce S-TunExSPEM, the SPEM 2.0 extension aimed at supporting the modelling as well as the exchange of safety-oriented processes. In particular, in Sect. 3.1 we focus on the modelling aspect and in Sect. 3.2 we focus on the exchangeability aspect.

3.1 Modelling Safety-oriented Information

In this subsection, we focus on one aspect of our SPEM 2.0 extension: its safety-tunability (recalled in the first part of its name *S-Tune*). Our extension involves mainly four SPEM 2.0 packages, namely Method Content, Process with Method, Managed Content and Process Structure.

To provide safety-tunability, we add an attribute to the Activity meta-class to allow process engineers to set the safety level. We only consider four levels since in case of negligible (e.g. no effect, level E in D0-178B) consequences related to the hazards, no specific safety-related process elements are needed. Moreover, we extend each meta-class pertaining to the definition of the core process elements (namely, RoleDefinition/etc. as depicted in Fig. 1) with a corresponding safety-related meta-class (SafetyRole/etc.). Similarly to what proposed for the core process elements-related meta-classes, we extend the Method Content Use-related meta-classes (recalled in Fig. 2) with corresponding safety-related meta-classes, as partially depicted in Fig. 4 (e.g. SafetyWorkProductUse, SafetyRoleUse, etc.). Finally, the extension of the WorkSequence meta-class permits process engineers to highlight safety-related flows within the process.

Whenever DO-178B provides information to further classify the core process elements, we add an attribute to the corresponding meta-classes to allow the kind to be set. For sake of clarity, in what follows we provide some examples. According to DO-178B, a safety activity (task) can be further characterized by setting its kind (check, review, or audit). Thus, as shown in Fig. 4, we add an attribute called *S-*

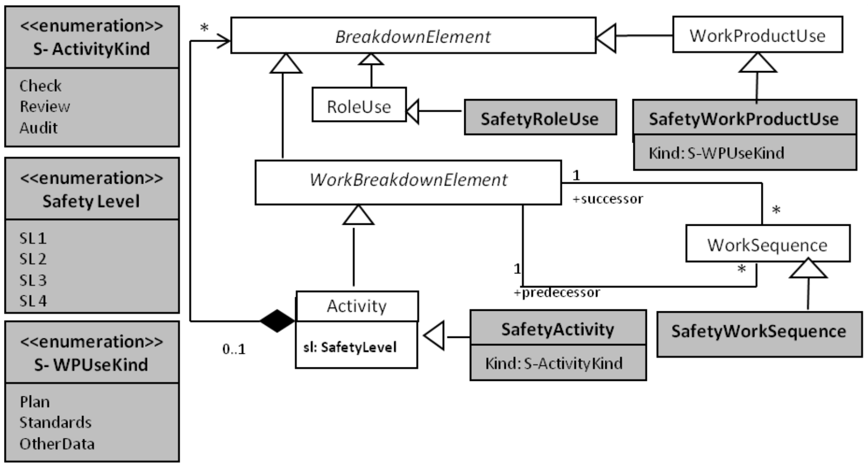


Fig. 4 Cut of S-TunExSPEM meta-model.

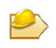
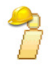
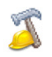
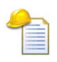
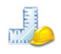
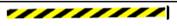
ActivityKind to the *SafetyActivity* class and an appropriate enumeration to allow the kind to be set. As seen in Sect. 2.2, workproducts that flow through the tasks belong to different kinds (Plans e.g. Software Development Plan, Standards e.g Software Design Standards, or other software life-cycle data e.g. code). Thus, an attribute is added to the SafetyWorkProduct meta-class and an additional enumeration is available to allow the kind to be set (Plan, Standards, OtherData). This characterization is possible also for Method Content Use-related meta-classes, as shown in Fig. 4 for SafetyWorkProductUse. DO-178B also allows guidances to be further characterized (namely, checklists to guide for example reviews, guidelines and additional supporting material). Thus, also in this case, even if not shown in Fig. 4, we add an attribute to the SafetyGuidance meta-class and an enumeration.

To the meta-classes, we associate intuitive icons. Table 2 shows some of the S-TunExSPEM icons to be used to model safety-related tasks, roles, tools, workproducts and guidelines. Except for the Safety Work Sequence, which is represented as a yellow/black line, the remaining elements are obtained by adding a safety hat to the original Method Content SPEM 2.0 icons presented in Table 1. Similarly, a safety hat is added for the Method Content Use SPEM 2.0 icons. According to the safety level, a different colour for the hat can be used (i.e. red for the most critical safety level, followed by orange, yellow and bitter lemon). In case of sub-processes related to non-safety functions, no hat is needed.

3.2 Exchangeability of safety-related processes

In this subsection, we focus on the other aspect of our SPEM 2.0 extension: its exchangeability (recalled in the second part of its name *Ex*). In particular, we present

Table 2 Graphical core elements of S-TunExSPEM

Task Definition	**Role Definition**	**Tool Definition**	**Work Product Definition**	**Guidance**
				
Safety Work Sequence				
				

the mapping between some S-TunExSPEM concepts and corresponding XPDL 2.2 concepts. We focus our attention on the safety-related concepts. The interested reader may find details concerning the entire mapping as well as a pseudo-code version of the transformation algorithm in [20]. The aim of this mapping is to support exchangeability of process models and thus enable the exploitation of engines (available off the shelf) for execution, simulation, monitoring purposes.

Table 3 shows our rather self-explanatory mapping which further develops what was presented in [10] to take into consideration the beneficial changes (introduced in XPDL 2.2), which allow for a better semantic mapping. As mentioned in Sect. 2.4, XPDL 2.2 provides modelling elements for the data/artefacts that flow within a process, thus instead of mapping a work-product onto an extended attribute as authors did in [10], we are able to map a work-product onto a closer semantic element. Similarly, we map the concept of guidance onto the concept of textual annotation. Moreover, we also preserve the distinction between RoleDefinition and RoleUse, by mapping these elements onto more appropriate XPDL 2.2 elements. We indeed map the reusable method content element role onto the concept of participant and we map the process-specific task-role (method content use element) onto the concept of lane. Then, to model the safety concern, we make an extensive usage of the extensibility mechanisms of XPDL.

Table 3 Concepts mapping

S-TunExSPEM	**XPDL 2.2**
SafetyRoleDefinition	Participant +extended attribute
SafetyTaskUse	Activity +extended attribute
SafetyWorkProductUse	DataObject+ extended attribute
SafetyRoleUse	Lane in a pool + extended attribute
SafetyGuidance	Annotation +extended attribute
SafetyTool	Application+extended attribute
SafetyWorkSequence	Transition + extended attribute. Remark: from-activity or to-activity must be an activity representing a SafetyTaskUse

4 Case Study

In this section, we show the usage of S-TunExSPEM by modelling the software development process defined in DO-178B, which was briefly recalled in natural language in Sect. 2.2. The purpose is not to provide a detailed model but to provide evidence with respect to the richer expressiveness of the language as well as its potential in terms of exchangeability. For sake of clarity, it must be highlighted that S-TunExSPEM only aims at offering usable and expressive modelling capabilities targeting safety-oriented processes. Its usage should allow process engineers to model safety concerns in a more straightforward way and to communicate with safety assessors more easily. S-TunExSPEM does not contribute to safer code directly. If the process mandated by the standard contributes to safer code and if this process is properly understood, S-TunExSPEM may help in spreading and formalizing its understanding as well as graphically recalling what should be done.

Fig. 5 shows the design phase modelled by using S-TunExSPEM. From the figure it is straightforward to grasp that this phase is dealing with some design decisions related to some safety concerns of major (yellow hat) relevance. Moreover, in case of need, it is straightforward to detect the roles that are responsible of safety related decisions. Hanna is the only human being in charge of the design. Hanna however has all the skills that are needed since she acts as safety expert as well as designer. Hanna is in charge of: checking that all the work-products in input are available, following the guidances and using the appropriate tools to provide all the work-products in output. It is also straightforward to identify safety-related work-products and thus be aware about the deliverables that are involved in the certification process.

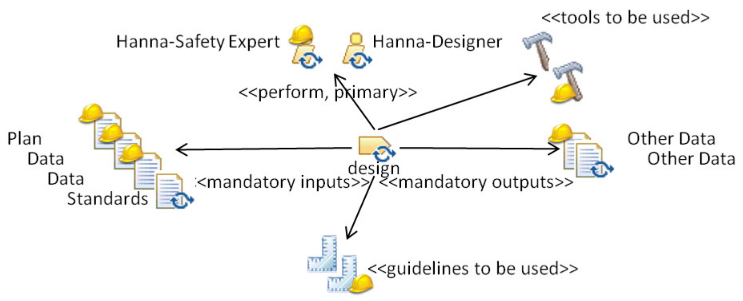


Fig. 5 DO-178B design phase in S-TunExSPEM.

Fig. 6 shows the dynamics of the entire software development process. For space reasons, however, in Fig. 6 we do not provide in S-TunExSPEM all the characteristics of the phases as done textually in Sect. 2.2 and graphically in Fig. 5 for the design phase. For the same reason, we do not show the usage of the safety-oriented flow that takes place whenever an output from the traceability check tasks is available as a feedback to the preceding task. Fig. 6 is simply aimed at showing that S-

TunExSPEM permits process engineers to intuitively separate safety concerns from functional concerns.

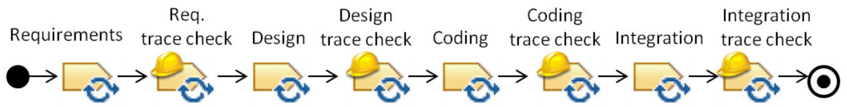


Fig. 6 DO-178B software development process in S-TunExSPEM.

In what follows, we provide the essential XPDL 2.2 snippets corresponding to some S-TunExSPEM process elements, depicted in Fig. 5. We do not provide the entire code but only significant parts needed to highlight our mapping related to safety concerns and our timely and pertinent exploitation of the current release of XPDL. In bold, we highlight the first-class entities for readability purposes.

```
<!-Input data of Activity (TaskUse) "Design" -->
<xpdl:Artifact ArtifactType="DataObject" FormalParameterRef="IN"
Id="DO1" Name="SW development Plan"></xpdl:Artifact>
<xpdl:Artifact ArtifactType="DataObject" FormalParameterRef="IN"
Id="DO2" Name="SW Requirements Data (functional)"></xpdl:Artifact>
<xpdl:Artifact ArtifactType="DataObject" FormalParameterRef="IN"
Id="DO3" Name="SW Requirements Data (Safety-related)"> </xpdl:Artifact>
<xpdl:Artifact ArtifactType="DataObject" FormalParameterRef="IN"
Id="DO4" Name="SW Design Standards"> </xpdl:Artifact>

<!-Output data of Activity (TaskUse) "Design" -->
<xpdl:Artifact ArtifactType="DataObject" FormalParameterRef="OUT"
Id="DO5" Name="SW design Description (Functional)"> </xpdl:Artifact>
<xpdl:Artifact ArtifactType="DataObject" FormalParameterRef="OUT"
Id="DO6" Name="SW design Description (Safety-related)"></xpdl:Artifact>
```

As the above snippets show, work-products involved in the design phase are defined as Artifacts of type DataObject. Moreover, if artefacts are provided in input (output, respectively), the attribute FormalParameterRef must be set to ”IN” (”OUT” respectively).

```
<!-Guidance attached to Activity (TaskUse) "Design" -->
<xpdl:Artifact ArtifactType="Annotation" Id="AN1" Name="Safety guidance">
</xpdl:Artifact>
<xpdl:Artifact ArtifactType="Annotation" Id="AN2" Name="guidance">
</xpdl:Artifact>
```

As the above snippets show, guidances involved in the design phase are defined as Artifacts of type Annotation.

```
<!-Participants  of Activity (TaskUse) "Design" -->
<xpdl:Participants>
<xpdl:Participant Id="RO1" Name="Designer"><xpdl:ParticipantType Type=""/)
<xpdl:Description>In charge of design decision related to functional
requirements </xpdl:Description></xpdl:Participant>
<xpdl:Participant Id="RO2" Name="Safety Expert"><xpdl:ParticipantType Type=""/)
<xpdl:Description>In charge of design decision
related to non-functional (safety) requirements
</xpdl:Description> </xpdl:Participant></xpdl:Participants>
```

As the above snippets show, the involved roles are defined as Participants.

```
<!-Pool and Lane containing TaskUse "Design" -->
<xpdl:Pools>
<xpdl:Pool BoundaryVisible="true" Id="RO1" MainPool="true"
Name="PARTICIPANT NAME" Orientation="HORIZONTAL" Process="SW Life Cycle">
<xpdl:Lanes>
<xpdl:Lane Id="" Name="Hanna">
<xpdl:Performers>
<xpdl:Performer>RO1</xpdl:Performer><xpdl:Performer>RO2</xpdl:Performer>
</xpdl:Performers>
</xpdl:Lane></xpdl:Lanes></xpdl:Pool></xpdl:Pools>
```

As the snippets concerning the pool specification states, Hanna, consistently with what modelled in Fig. 5, is the only human being in charge of the design. Hanna is the actual role responsible of acting as designer as well as safety expert.

```
<!-extended attributes for the safety-oriented customization  -->
<xpdl:ExtendedAttributes>
<xpdl:ExtendedAttribute Name="Safety Role" Value="Safety Expert">
<xpdl:ExtendedAttribute Name="Safety Data object" Value="SW Development Plan">
<xpdl:ExtendedAttribute Name="Safety Data object"
Value= "SW Requirements Data (Safety-related)">
<xpdl:ExtendedAttribute Name="Safety Data object" Value="SW Design Standards">
<xpdl:ExtendedAttribute Name="Safety Data object"
Value= "SW Design Description (Safety-related)">
<xpdl:ExtendedAttribute Name="Safety Guidelines" Value="Safety Guidance">
</xpdl:ExtendedAttributes>
```

As the above snippets show, extended attributes customize/specialize the XPDL 2.2 concepts towards safety. As presented in Table 3, an extended attribute is used to customize each safety-related process element.

5 Related Work

In this section, we discuss those related works that contribute to either provide modelling capabilities for safety-oriented processes or transform process models into other models for execution purposes. To support the modelling of safety-oriented processes, a new meta-model, called Repository-Centric Process Metamodel is provided in [13, 29, 14]. Besides, meta-classes aimed at representing generic process concepts (e.g. activity), RCPM includes one safety related meta-class (check point), which specializes a generic meta-class. RCPM also includes one meta-class to represent safety-related relationships (safety relationship). Finally, a safety level can be specified for a process. Thus, in principle, safety process engineers are enabled to model safety-related activities and how these activities are related from a safety-related flow point of view.

Similarly to what is proposed in [13, 29, 14], we also provide a meta-class to represent safety-related activities as well as a meta-class to represent safety-related flows. However, our work highly differs from [13, 29, 14] since we do not introduce a new meta-model but propose to extend an existing one. Moreover, we broaden our focus on other conceptual elements that are crucial in the context of safety critical systems development. The concept of role, for instance is of paramount importance to stress that every piece of information produced during the development process requires the appropriate set of skills. Similarly, the way in which an activity is performed is of paramount importance. Thus guidelines represent first-class modelling elements. Finally, we also propose a rather intuitive safety-oriented concrete syntax.

Another related work which was aimed at modelling DO178B processes by using OpenUp is presented in [6]. This work is of interest for its pioneering intention of exploiting existing process modelling capabilities to document safety-related processes. Authors conclude that customization of the existing capabilities is needed.

When quality attributes (e.g. safety) are crucial for the systems development, it becomes relevant to model the techniques that target that attribute. In [7], authors investigate how safety analysis techniques could be modelled in SPEM. They explore two alternatives: the usage of *step* eventually combined with *guidance* or the usage of *task* eventually combined with *guidance*. In our case, we also model the techniques but we only use *guidance* since we model the remaining and conceptually different information onto other modelling elements.

Concerning process models interchange or simulation/execution/monitoring, several works exist. Some of these works have investigated approaches for mapping process models onto interchangeable models others have provided SPEM 2.0 extensions to enhance its support for executability.

In [10], authors provide a mapping as well as a transformation algorithm to transform SPEM1.0 models into XPDL (draft 1.0) models. As a running example they use a review process. As mentioned in Sect. 3.2, our approach borrows from this one and goes beyond it since we transform S-TunExSPEM models into XPDL 2.2 models and thus we provide support for safety concerns and a more suitable semantic mapping. In [3], authors make a critical analysis of SPEM 2.0 support for executability and then propose a SPEM 2.0 extension, called xSPEM. Their extension includes a set of concepts and behavioural semantics aimed at enhancing SPEM 2.0 executability. Similarly, in [8, 9], authors propose a tool-supported SPEM 2.0 extension, called eSPEM to enhance the support for executability. eSPEM is defined as CMOF meta-model and is based on both SPEM 2.0 and UML Superstructure. Authors replace the Process Behaviour package recalled in Sect. 2.3 with a new one defining fine-grain concepts for behaviour specification.

To provide our contribution, we have focused our attention on the textual descriptions of safety-related processes available in safety standards. We have not yet tried to model real processes and thus the mechanisms for behavioural specification, provided within the SPEM 2.0 Process Behaviour package, were enough for our purposes. So, we have not integrated the above extensions within our proposal.

6 Conclusion and Future Work

To ensure the safety of safety-critical systems, compulsory as well as advisory safety standards have been issued. Some of these standards define (prescriptive) safety-oriented processes. Modelling processes in compliance with the standards is relevant to provide process-based evidence for certification purposes. To support safety-oriented process engineers in these activities, in this paper we have proposed a PML, called S-TunExSPEM, obtained by extending SPEM 2.0 with safety-specific constructs extracted by examing safety standards (mainly DO-178B). Moreover, besides offering modelling capabilities for safety-related concepts, S-TunExSPEM provides the first tile to pave the road towards process models exchangeability aimed at exploiting existing simulation, monitoring and execution engines.

In the immediate future, first of all, we aim at validating the effectiveness of our proposal in supporting process modelling activities in industrial settings. We are currently in contact with some military as well as civil organizations responsible for software development of avionics software. Then, we aim at investigating model transformation approaches to automatize the generation of XPDL 2.2 models from S-TunExSPEM models. In a long-term future, we plan to provide a tool-chain support for modelling and monitoring / executing / etc. safety-processes.

Finally, since safety-oriented processes can be considered as a process line [12], safety-related process elements of S-TunExSPEM could be considered as variability elements and divided into commonalities, partial commonalities, and variabilities either by reusing the current SPEM 2.0 support for variability modelling or by adopting the in-progress SPEM 2.0 extension for process lines, called vSPEM [17]. The intention would be to contribute to pushing towards a SPEM 3.0 version allowing for richer modelling support as well as exchangeability/execution targeting safety.

Acknowledgements This work has been partially supported by the European Project ARTEMIS SafeCer [2] and by the Swedish SSF SYNOPSIS project [26].

References

1. Acuña, S.T., Ferré, X.: Software Process Modelling. In: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL, pp. 237–242 (2001)
2. ARTEMIS-JU-269265: SafeCer-Safety Certification of Software-Intensive Systems with Reusable Components. http://www.safecer.eu/ (2013)
3. Bendraou, R., Combemale, B., Cregut, X., Gervais, M.P.: Definition of an Executable SPEM 2.0. In: Proceedings of the 14th Asia-Pacific Software Engineering Conference, Nagoya, Japan, APSEC, pp. 390–397 (2007)
4. Bendraou, R., Jezequel, J., Gervais, M.P., Blanc, X.: A Comparison of Six UML-Based Languages for Software Process Modeling. Software Engineering, IEEE Transactions **36**, 662–675 (2010)
5. Berk, R.H.: An Analysis of Current Guidance in Certification of Airborne Software. Master's thesis, Massachusetts Institute of Technology, Cambridge, USA (2009)
6. Bertrand, C., Fuhrman, C.P.: Towards Defining Software Development Processes in DO-178B with Openup. In: Proceedings of 21st IEEE Canadian Conference on Electrical and Computer Engineering, CCECE, pp. 851–854. Niagara Falls, Ontario, Canada (2008)
7. Chiam, Y.K., Staples, M., Zhu, L.: Representation of Quality Attribute Techniques Using SPEM and EPF Composer. In: European Software Process Improvement, EuroSPI. Springer, Spain (2009)
8. Ellner, R., Al-Hilank, S., Drexler, J., Jung, M., Kips, D., Philippsen, M.: eSPEM-A SPEM Extension for Enactable Behavior Modeling. In: Proceedings of 6th European Conference Model Foundation and Appliation, Paris, France, ECFMA, pp. 116–131. Springer (2010)

9. Ellner, R., Al-Hilank, S., Jung, M., Kips, D., Philippsen, M.: Integrated Tool Chain for Meta-model-based Process Modelling and Execution. In: Proceedings of First Workshop on Academics Modeling with Eclipse, Lyngby, Denmark, ACME (2012)
10. Feng, Y., Mingshu, L., Zhigang, W.: SPEM2XPDL-Towards SPEM Model Enactment. Software Engineering. Front. Comput. Sci. China, Higher Education Press, Bejing, China pp. 1–11 (2008). Co-published with Springer-Verlag GmbH
11. Fuggetta, A.: Software Process: A Roadmap. In: Proceedings of the International Conference on Software Engineering, New York, USA, ICSE, pp. 25–34 (2000)
12. Gallina, B., Sljivo, I., Jaradat, O.: Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification. In: Post-proceedings of the 35th IEEE Software Engineering Workshop, SEW-35. Greece (2012)
13. Hamid, B., Geisel, J., Ziani, A., Gonzalez, D.: Safety Lifecycle Development Process Modelling for Embedded Systems - Example of Railway Domain. In: Proceedings of Software Engineering for Resilient Systems, SERENE, pp. 63–75. Pisa, Italy (2012)
14. Hamid, B., Zhang, Y., Geisel, J., Gonzalez, D.: First Experiment on Modeling Safety Life-Cycle Process in Railway Systems. International Journal of Dependable and Trustworthy Information Systems, IGI Global, Hershey - USA **2**, 17–39 (2011)
15. Health and Safety Executive (HSE): Out of Control. Why Control Systems Go Wrong and How to Prevent Failure (2003)
16. Jackson, D., Thomas, M., Limmet, L.I.: Software for Dependable Systems: Sufficient Evidence? National Academy Press, Washington DC, USA (2007)
17. Martínez-Ruiz, T., García, F., Piattini, M., Münch, J.: Modeling Software Process Variability: An Empirical Study. IET Software **5**, 172–187 (2011)
18. Object Management Group: Software & Systems Process Engineering Meta-Model (SPEM), v2.0. Full Specification formal/08-04-01 (2008)
19. Panesar-Walawege, R.K., Sabetzadeh, M., Briand, L.: Using Model-Driven Engineering for Managing Safety Evidence: Challenges, Vision and Experience. In: Proceedings of the 1st International Workshop on Software Certification, WoSoCER, pp. 7–12. Hiroshima, Japan (2011)
20. Pitchai, K.R.: An Executable Meta-model for Safety-oriented Software and Systems Development Processes within the Avionics Domain in Compliance with RTCA DO-178B. Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, Sweden (2013)
21. Redmill, F.: Safety Integrity Levels - Theory and Problems. Lessons in System Safety. In: Proceedings of the Eighth Safety-critical Systems Symposium, Southampton (2000)
22. RTCA Inc: Software Considerations in Airborne Systems and Equipment Certification, RTCA DO-178B (EUROCAE ED-12B). Washington DC (1992)
23. Ruiz-Rube, I., Dodero, J.M., Palomo-Duarte, M., Ruiz, M., Gawn, D.: Uses and Applications of SPEM Process Models. A Systematic Mapping Study. Journal of Software Maintenance and Evolution: Research and Practice pp. 1–32 (2012)
24. Rushby, J.: New Challenges in Certification for Aircraft Software. In: Proceedings of the ninth ACM International Conference on Embedded software, New York, USA, EMSOFT, pp. 211–218 (2011)
25. Shapiro, R.M.: XPDL 2.2: Incorporating BPMN2.0 Process Modeling Extensions. Extracted from BPM and Workflow Handbook, Future Strategies (2010)
26. SYNOPSIS-SSF-RIT10-0070: Safety Analysis for Predictable Software Intensive Systems. Swedish Foundation for Strategic Research
27. Workflow Management Coalition: Workflow Management Coalition Workflow Standard- Process Definition Interface - XML Process Definition Language, WfMC-TC-1025, v2.2, (2012)
28. Zamli, K.Z., Lee, P.A.: Taxonomy of Process Modeling Languages. In: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, AICCSA, pp. 435–437 (2001)
29. Zhang, Y., Hamid, B., Gouteux, D.: A metamodel for representing safety lifecycle development process. In: Proceedings of the Sixth International Conference on Software Engineering Advances (ICSEA), IEEE Computer Society press, Barcelona, Spain, pp. 550–556 (2011)