# Ontology-based Identification of Commonalities and Variabilities among Safety Processes

Barbara Gallina[1] and Zoltán Szatmári[2]

[1] Mälardalen University, Västerås, Sweden barbara.gallina@mdh.se
[2] Resiltech Srl, Pontedera, Italy szatmari@mit.bme.hu

**Abstract.** Safety standards impose requirements on the process used to develop safety-critical systems. For certification purposes, manufacturers have to properly interpret and meet these requirements, which exhibit commonalities and variabilities. However, since different terms are used to state them, the comparative work aimed at manually identifying and managing these commonalities and variabilities is hard, time-consuming, and costly. In this paper, we propose to solve this problem by creating ontology-based models of safety standards and automate the comparative work. Then, we show how the result of this comparative study can be exploited to semi-automate the generation of safety-oriented process line models. To illustrate our solution, we apply it to portions of ISO 26262 and EN 50126. Finally, we draw our conclusions and future work.

## 1 Introduction

Safety standards impose requirements on the development process of safety-critical (software) systems. For certification/conformance purposes, manufacturers have to properly interpret and meet these requirements, which exhibit commonalities and variabilities. More specifically, commonalities and variabilities can be identified when comparing different criticality levels within the same version of a single standard, different versions of the same standards, or different standards within the same domain or even within different domains. The time and cost required for performing the comparative work increases when moving from one single version to different standards within different domains. This is due to the usage of different terms, which sometimes do not denote a different semantics. Irrelevant terminological differences are sometimes introduced for political reasons [1]. These differences slow down not only the provision of deliverables but also the audit of such deliverables. Identifying commonalities and variabilities is crucial to enable manufacturers to speed up the creation of process-related deliverables via systematic reuse. At the same time well-defined and managed reuse, speed up the audit process on the certification authority side. In the context of security-informed safety [2], irrelevant terminological differences contained within safety-specific and security-specific standards prevent cross-fertilization as well as reuse. Authors state: The commonalities between safety and security

are frequently obscured by the use of different concepts and terminologies. Indeed, there is considerable variation in terminology both within and between the safety and security communities. Thus, to achieve a shared understanding of the key concepts within each domain, there is a need to establish a lingua franca or even a common ontology [2]. In this paper, to ease the identification and systematization of commonalities and variabilities, we propose a new method called OPER, which stands of Ontology-based Process Elements Reuse. In our method, we propose to provide ontology-based models (given in compliance with OWL2.0) related to the safety processes mandated within the standards, then to semi-automate the identification of commonalities and variabilities. Finally, based on model-driven engineering principles, we propose to semi-automate the generation of Safety-oriented Process Line (SoPL) [3] models (given in compliance with SPEM (Software Process Engineering Meta-model) 2.0) based on the calculated commonalities and variabilities. OPER supports the creation of a lingua franca but at the same time allows certification bodies to preserve their specificities if this is required. The rationale behind OPER is that ontologies are able to capture domain knowledge in a precise way. Ontologies provide a natural formalism for representing domain knowledge and capturing constraints. In this paper, we use ontologies to capture the process (and SoPL) requirements. Then, we apply ontology-related reasoning to manipulate, validate the constructed models or transform them to a required representation. To show the usage and effectiveness of OPER, we apply it on small portions of safety standards. The rest of the paper is organized as follows. We present: essential background, in Section 2; OPER, in Section 3; OPER's application, in Section 4; finally, conclusion and future work, in Section 5.

## 2 Background and related work

**Safety standards** (focus on ISO 26262 [4] and EN 50126 [5]). Based on Gallina et al. [3], for both standards, we focus on a specific portion of the process that includes hazard analysis and risk assessment (HARA) activities. The portion is named Concept phase in ISO 26262 and Risk Analysis in EN 50126. For the HARA activities, we recall the required information that is necessary to understand our examples presented in Section 4. *ISO 26262-HARA clause* is aimed at: identifying and categorizing the hazards; formulating the safety goals related to the preventions/mitigations of the hazards. This clause consists of a number of tasks that need to be performed in a specific order: 1) Initiation of HARA, 2) Situation analysis, 3) Hazard identification, 4) Hazard classification, 5) ASIL determination, 6) Determination of safety goals, and 7) Verification of hazard analysis, risk assessment, and safety goals. *EN 50126-Risk Analysis Phase* is aimed at: empirically or creatively identifying the hazards associated with the system; estimating the risk associated with the hazards; developing a process for risk management. This phase consists of a number of tasks that need to be performed in a specific order: 1) Hazard identification, 2) Hazard classification, 3) Risk evaluation, 4) Determination and classification of acceptability of the

risk, 5) Establishment of the Hazard Log, 6) Assessment of all phase's tasks. **Safety-oriented Process Lines**-A *Safety-oriented Process Line* (SoPL) [3] is a family of highly related safety-oriented processes that are built from a set of core process assets in a pre-established fashion. Core assets can be classified as full or partial commonalities or variabilities [3]. A *partial commonality* denotes a composite process element (e.g., a task composed of steps) that contains a subset, which constitutes the commonality among all the composite process elements of the same type. For instance, two tasks represent a partial commonality if they contain a subset of equal steps. During the domain engineering phase [3], safety processes are compared to retrieve core assets: (partial) commonalities and variabilities. Once the core assets are defined, a safety process can be derived by performing two steps: 1) selection of all the (partial) commonalities plus the desired variants at variation points; 2) composition of the selected elements. **SPEM 2.0**-SPEM 2.0 [6] is the OMG's standard for systems and software process modelling. SPEM 2.0 offers support for modeling reusable process content as well as process variability. In SPEM 2.0, a process element (e.g. an activity) can be defined as a variability element and its variability type can be characterized. The *Variability Type* enumeration class defines the different types of variability. In this paper, we only recall one variability type, namely *contributes*, which logically replaces the original element (the base) with an augmented variant. In SPEM 2.0, the expected work can be broken down hierarchically via a series of elements (e.g., activities). **Ontology-related Concepts**-An *ontology* [7] is a model that represents a domain and is used to reason about the inter-related objects in that domain. An ontology generally includes: 1) Individuals (Objects) that are basic elements of the domain. 2) Classes that are sets of objects sharing certain characteristics. 3) Relations (properties) that are sets of pairs (tuples) of objects. Relations define ways in which objects can be associated to each other. 4) Attributes that are special relations where the class is related to a concrete domain (e.g. integer, string). To automate the analysis of an ontology (i.e., inference of logical consequences from a set of asserted facts or axioms and evaluation of model consistency), reasoners are used. Reasoners are also used to check whether a class is a subclass of another class(subsumption test). By performing such tests it is possible to compute the inferred ontology class-hierarchy, i.e., a class-hierarchy. Ontologies can be easily extended and combined. OWL2.0 (Web Ontology Language) [7] is an ontology language. An OWL2.0 ontology consists of a collection of facts, annotations, and axioms, which describe different items (individuals, concepts, relations and attributes). OWL2.0 can operate with different expression levels, called OWL2.0 profiles, which allow different sets of axioms. In this paper, we choose OWL2.0 EL, because it allows us to define: Subclass, Disjoint classes, Disjoint union and Equivalent Classes. A commonly used ontology development tool is Protégé [8] since it facilitates the use of several reasoners and provides application program interfaces (e.g., OWLAPI) for efficiently querying/manipulating the dataset, generated as the output of the reasoning. **Model-driven Engineering (MDE)**-MDE is a model-centric software development methodology. Model transformations are used to refine

models. A model transformation (e.g. Model-to-Model), defined as a set of rules, transforms a source model (compliant with one meta-model) into a target model compliant with the same or a different meta-model. **Related Work**-No related work exists on ontology-based identification of commonalities and variabilities among processes. As already extensively explained by Gallina et al. [3], SoPL is an extension of the *process line* notion.

## 3 OPER

OPER builds on top of previous work and combines principles related to ontologies, SoPL engineering, and MDE. OPER is constituted of three chained tasks, which are: T1 (Ontology-based safety process modeling), T2 (Ontology-based Commonalities & Variabilities Identification and Merging), and T3 (SPEM2.0-compliant SoPL model generation). **In T1**, a process engineer in cooperation with an ontology and a standards expert is responsible of modeling safety processes according to the best practices in ontology modeling (i.e., OWL2.0 EL). These models are also based on the SPEM 2.0-terminology. For instance, the structures that represent the breaking down of the work (e.g., process, phases, activities, tasks, etc.) are aligned. To model the processes, Protégé is used. To provide such models, we map SPEM 2.0 and OWL2.0 EL concepts. The mapping, shown in Table 1, focuses on concepts related to the process structure.

**Table 1.** Concepts mapping

| **SPEM2.0** | BreakDownElement | Variability type | Equivalence relation | Composition of BreakDownElements |
|---|---|---|---|---|
| **OWL2.0 EL** | Class | ObjectProperty | EquivalentClasses Axiom | ObjectProperty |

This structure describes a hierarchy of process elements, where phases are hierarchically broken down into activities/tasks/steps. Each process is represented as a tree (according to the graph theory terminology). We interpret the full and the partial commonality properties on two process elements in this hierarchy. The process elements are mapped to classes in the ontology, and the relations are expressed using object properties. For sake of clarity, we point out that we only consider a two-level hierarchy of work decomposition. **In T2**, the experts identify commonalities and variabilities in order to merge them within a single model representing an ontology-based SoPL. The trivial equivalences between corresponding process elements are defined. More precisely, an "equivalent of" axiom is added to the model (including the two safety process models) when two safety process elements are called in a different way in the different standards but they denote the same concept [9]. Based on the definition of full (or partial) commonality and the previously defined matching, our bottom-up algorithm identifies the safety process elements that have some type of commonality. Our algorithm is implemented in Java and uses OWLAPI. The first part of our implementation consists of an algorithm aimed at constructing the common subtree, where the nodes are the process elements and the edges are the refinement relations between them. This algorithm traverses the safety process

models (trees) and based on the defined equivalence relations the commonalities are identified. In order to get the SoPL model, the variabilities should be also added to the model. In the second phase the algorithm traverses each process model and identifies the variabilities and adds the required process elements to the process line model and connects them using the extends relation to the required model element. In our implementation the safety process-related trees are defined via a recursive data structure as presented in Table 2, column-1. Each node in the tree is an object that has a parent and that can be related to other nodes in the tree, via *composes* and *contributes* relations.

**Table 2.** Pseudo code

| **Recursice data structure** | **Common subtree construction function** |
|---|---|
| Structure TreeNode { | function buildCommonSubTree(TreeNode: A,B,C){ |
| TreeNode: parent | foreach (nodeA=A->composes) |
| TreeNode[]: composes | if (hasEquivalent(nodeA,B->composes) |
| TreeNode[]: contributes } | nodeB=getEquivalent(nodeA,B->composes); D = new TreeNode(nodeA,nodeB); |
| | buildCommonSubTree(nodeA,nodeB,D) } |

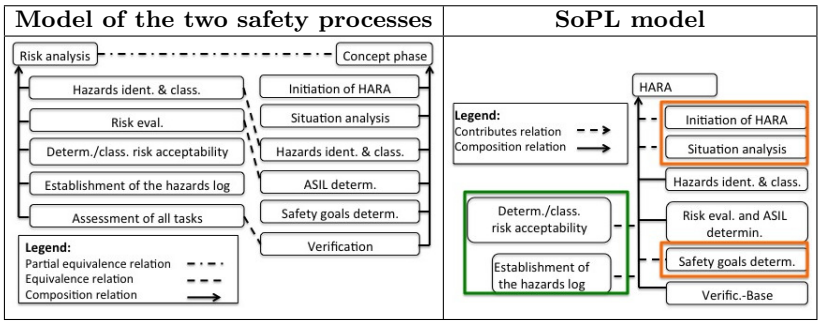
The pseudo code given in in Table 2, column-2, represents a recursive function that is used to build a common subtree. The function is called with three process trees as parameters: the root elements (A and B) and a newly created root element of the common process tree (C). In each recursive call A, B, C logically represent the same node (same hierarchical level and process element). For each child tree nodes connected by a composes relation to the tree node represented by A we identify the equivalent pair in the set of child nodes of B. After a successful match, we add a new node (D) (based on the two equivalent nodes (nodeA and nodeB)) as a child node of C to the common subtree and start a new recursive iteration. **In T3**, the process engineer jointly with an ontology expert generates a SPEM2.0-compliant model representing the SoPL by using a model transformation implemented within a transformation engine. During this task, we define a transformation aimed at generating a SPEM 2.0-compliant SoPL model from an OWL2.0-compliant SoPL model. Our transformation (still part of an ongoing work) includes the following rules: 1) Identify the hierarchical levels of the process tree. The leaves in the ontology-based process tree are mapped onto the lowest level of the SPEM2.0 process structure. Then, by parsing the tree bottom-up, each parent node in the process tree is mapped to the next level of the SPEM2.0-compliant work breakdown structure. 2) Identify the base elements of the ontology-based SoPL: determine the root process element and by following the composes relation the common subtree can be identified. 3) Transform the base into SPEM 2.0 SoPL model: each process element in the common subtree should be transformed into a SPEM 2.0 work breakdown element based on the hierarchical level identified in the first rule. 4) Transform the variability-related part of the SoPL model. To do that, the following steps should be performed: 1) traverse the process tree by starting from the root process element; 2) follow both the composes and contributes relations. Every process element that is

characterized by a contributes relation should be transformed into a variability element and its variability type attribute should be set to contributes.

## 4 Applying OPER

We construct the two ontologies that represent the safety processes. For each standard, we consider only one clause, see Section 2, (interpreted as SPEM2.0-task) and we only model one hierarchical level. The two safety process trees are depicted in Table 3, column-1, on the left-hand side, we can see the model of the EN 50126-*Risk Analysis* phase and on the right hand side the ISO 26262-*Concept phase*. Before executing the algorithm to create the SoPL model, we add the (partial) equivalence relations. In Table 3, column-1, these relations are shown by using dotted lines. After the execution of our algorithm, we obtain the SoPL ontology, depicted in Table 3, column-2. *HARA* represents a partial commonality. The naming of the new nodes is performed semi-automatically. First, if two nodes are merged into one single node, the names are concatenated automatically. Then, a manual post-processing performed in order to provide a human-readable name. For presentation purposes, in Table 3, column-2, we show the result of post-processing. Simplified names for the common process elements instead of the generated ones are given. In Table 3, column-2, we present the commonalities (via *composes*) and the variabilities (via *contributes*).

**Table 3.** Ontologies models




This ontology is built up from three parts using the ontology composition support: 1) the commonalities, the variabilities that are derived from 2) EN 50126 (marked in green) and from 3) ISO 26262 (marked in orange). By applying the transformation rules given in Section 3, we can manually create the SoPL model. Based on the model depicted in Table 3, column-2, the hierarchical level of the process elements can be specified and afterwards the base (common subtree) can be mapped to SPEM 2.0 SoPL model. The variabilities (marked in orange and green) are mapped to work breakdown elements and the contributes relation is used to connect them to the required place in the model. The base, Base-Task in Fig. 1, can vary in an additive way via the contributes relationship to distinguish tasks that are compliant with either EN 50126 or ISO 26262.


**Fig. 1.** Partial safety-oriented task line.

## 5 Conclusion and Future Work

To reduce the complexity, cost, and time related to the interpretation and comparison of standards, in this paper, we presented OPER, a novel method that permits users to: 1) refer to a common process-related lingua franca, 2) semi-automate the standards comparison, and 3) generate SoPL models from safety process models represented via ontologies. The method was presented in the context of safety standards. However, more in general, it is applicable to normative documents that contain process-related requirements. In this paper, we focused on simple process structures. In a short-term future, we will tackle more complex structures. In cooperation with industry and assessors, we will properly define the concepts mapping that underlies the automated comparative work. In a medium/long-term future, we plan to provide a prototype of tool-chain (including THRUST [10]), aimed at providing evidence concerning the effectiveness in terms of time and cost reduction (manual vs. semi-automatic work).

**Acknowledgments.** This work is supported by the Swedish Foundation for Strategic Research (SSF) project SYNOPSIS-SSF-RIT10-0070.

## References

1. Ferrell, T., Ferrell, U.: Assuring avionics-updating the approach for the 21st century. In: Computer Safety, Reliability, and Security. Volume 8696 of LNSC. Springer (2014) 375–383
2. Bloomfield, R., Netkachova, K., Stroud, R.: Security-informed safety: If it's not secure, it's not safe. In: Software Engineering for Resilient Systems. Volume 8166 of LNSC. Springer (2013) 17–32
3. 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 Software Engineering Workshop (SEW), IEEE (Oct. 2012)
4. ISO26262: Road vehicles Functional safety. International Standard (2011)
5. BS EN50126: Railway applications: The specification and demonstration of Reliability. Availability, Maintainability and Safety (RAMS) (1999)
6. Object Management Group: Software & Systems Process Engineering Meta-Model (SPEM), v2.0. Full Specification formal/08-04-01. (2008)
7. OWL 2 Web Ontology Language: http://www.w3.org/tr/owl2-syntax/
8. Protégé: http://protege.stanford.edu/
9. Pataricza, A., Gönczy, L., Kövi, A., Szatmári, Z.: A methodology for standards-driven metamodel fusion. In: Model and Data Engineering. Volume 6918 of LNSC. Springer (2011) 270–277
10. Gallina, B., Lundqvist, K., Forsberg, K.: THRUST: A Method for Speeding Up the Creation of Process-related Deliverables. In: Proceedings of the 33rd IEEE Digital Avionics Systems Conference. DASC (2014)