Modeling a Safety- and Automotive-oriented Process Line to Enable Reuse and Flexible Process Derivation

Barbara Gallina and Shaghayegh Kashiyarandi
IDT, Mälardalen University,
Västerås, Sweden

Helmut Martin and Robert Bramberger
VIRTUAL VEHICLE Research Center,
Graz, Austria

***Abstract*—ISO 26262 is a recently introduced automotive functional safety standard. This standard imposes new requirements that must be fulfilled for conformance purposes. Thus, companies used to develop safety-related E/E systems in compliance with either only Automotive SPICE (ASPICE) or a combination of ASPICE and IEC 61508 have to quickly perform a gap analysis in order to introduce adequate changes in their way of working. Implementing such changes in a visionary way with expectations of a long-term payback is an urgent open issue. To contribute to addressing such issue, in this paper, we introduce a safety-oriented process line-based methodological framework to model commonalities and variabilities (changes) between the standards to enable reuse and flexible process derivation. To show the usefulness of our approach, we apply it to model a process-phase line for the development of safety-critical control units. Finally, we provide our lessons learned and concluding remarks.**

Keywords—Automotive SPICE; IEC 61508; ISO 26262; safety processes; safety-oriented process lines; process line modeling.

I. Introduction

In the context of safety-critical automotive systems engineering, various standards (e.g. ASPICE [1] and IEC 61508 [2]) play a crucial role in prescribing process reference models, which in some cases overlap and thus exhibit several similarities. More recently, ISO 26262 [3], which is a new standard for functional safety and which represents the automotive specialization of IEC 61508, has entered the scene with new requirements on the development process. As a consequence, since compliance with the process reference model may constitute a mandatory requirement for certification purposes, companies used to develop safety-related E/E systems in compliance with either only ASPICE or a combination of ASPICE and IEC 61508 have to quickly perform a gap analysis in order to introduce adequate changes in their processes. Via a gap analysis ad-hoc adjustments can be performed. Since however a gap analysis represents a timing window during which process engineers have to identify what changes and what remains the same, we propose to systematize this effort by implementing the safety-oriented process line approach, which was explored in [5] in the framework of the SafeCer [6] project. We thus propose to systematically model process elements as either commonalities or variabilities in order to enable reuse and flexible process derivation. The relevance of the adoption of a safety-oriented process line approach is motivated also by the fact that process reference models included in the standards allow flexible but thoroughly justified interpretations and customizations, which can be modelled as variabilities.

Thus, in this paper, to enable process engineers to implement changes with expectations of long-term payback, we introduce a methodological framework to model the commonalities and the variabilities that exist between automotive standards as well as the variabilities stemming from company-specific as well as project-specific interpretation and customization. Our methodological modelling framework uses the SPEM (Software Process Engineering Meta-model) 2.0 [8] process modeling language, which is implemented within the process modeling tool EPF-Composer [9]. To show the usefulness of our approach, we apply our framework to model a process-phase line for the development of safety-critical control units.

The rest of the paper is organized as follows. In Section II, we provide essential background information. In Section III we present our safety-oriented process line-based methodological modeling framework and its application. In Section IV, we draw some lessons learned. In Section V we discuss related work. Finally, in Section VI we present some concluding remarks and future work.

II. Background

In this section, we briefly present the background on which we base our work. In particular, in Section II.A we provide essential information concerning the system design phase of the automotive standards under examination. In Section II.B, we recall safety-oriented process line concepts and guidelines and, finally, in Section II.C we recall how processes and their variations can be modelled in SPEM2.0/EPF.

A. System design phase in the examined automotive standards

In this subsection, we focus on three standards, namely quality development standard ASPICE and two safety-related standards ISO 26262, and IEC 61508. The rationale behind the selection of these three standards is that they represent different but overlapping intra-domain (namely, automotive and industry) standards. For each standard, we provide a brief overview and then we focus on the system design phase. For this phase, we provide pointers to the normative parts, which are necessary to fully understand the application of our safety-oriented process line-based approach presented in Section III.

IEC 61508 deals with generic functional safety and is intended as a basis safety standard, applicable to different domains. ISO 26262 is one derivation of the IEC 61508. These two standards provide prescriptive development processes for achieving functional safety by reducing the risk of systematic failures. ISO 26262 and IEC 61508 include quality requirements that are in common with ASPICE, which focuses on process improvement. ASPICE provides a Process Reference Model (PRM). The PRM is composed of activities (in ASPICE named *Base Practices*), which are also covered by ISO 26262 and IEC 61508.

In ASPICE the majority of the activities concerning system design is part of the process *ENG.3 System architectural design*. Within this process the *Process Purpose*, the *Process Outcomes* and *Output Work Products* are defined. The structure of IEC 61508 differs significantly from ASPICE. IEC 61508 does not describe what should be done but sets out objectives and requirements for the activities. The system design phase is covered mainly in chapter 7.2 *E/E/PE system design requirements specification* and 7.4 *E/E/PE system design and development* of part two. Work products are not mentioned in the normative part. Finally, ISO 26262 part 4 – chapter 7 covers system design (entitled *System design*). This chapter includes objectives, input from other activities, requirements for the activities and work products.

B. Safety-oriented process line: concepts and guidelines

A *process line* [4] is a family of highly related processes that are built from a set of core process assets in a pre-established fashion. A *safety-oriented process line* is a process line that targets safety processes [5]. A (safety-oriented) process line approach is constituted of three phases: scoping (i.e. definition of the set of processes to be examined as a family), domain engineering (i.e. commonalities and variabilities identification and modeling), process engineering (single processes modelling via selection and composition of reusable commonalities and variabilities). Comparisons among safety processes characterize the main activity of the domain engineering phase. Through comparisons, it is possible to identify what can vary (variabilities) between safety processes and what, instead, remains unchanged (commonalities). At a first glance, processes defined in different standards seem to exhibit only variabilities. Terminological differences constitute a barrier to a straightforward identification of commonalities. Moreover, processes are constituted of phases, which in turns are constituted of a set of activities, which in turn are constituted of a set of tasks and which, finally, in turns are constituted of a set of steps. Thus, commonalities are unlikely at the root level of this nested structure. From an execution point of view, phases, activities, tasks, etc. may be performed in a different order. From a pure syntactical comparison, all these differences may be interpreted as variabilities. However, to be able to justify a process line approach the amount of commonalities must be greater than the amount of variabilities. To reduce the variabilities and increase the commonalities, two definitions are at disposal:

Partial commonality: whenever process elements of the same type (e.g. tasks) expose at least one common aspect (e.g. at least a step is equivalent).

In this paper, this definition is used in a flexible way. When comparing process elements of the same type, either the entire set of processes (process line) is considered or subsets of them. More specifically, the heterogeneous set of standards examined in this paper is divided into two subsets: one containing the non-safety-related standard (ASPICE) and the second containing the safety-related standards (ISO 26262 and IEC 61508). This flexible usage provides the potential to create a greater extent of reusable process elements.

Full commonality whenever process elements of the same type (e.g. tasks) expose only common aspects (e.g. all steps are equivalent).

For the sake of terminological completeness, we also clarify that a process variant is a representation of a particular instance of a variable process element of the real world or a variable property of such an element.

C. SPEM2.0 and Eclipse Process Framework

SPEM 2.0 [8] is the OMG (Object Management Group)'s standard for software process modelling. In the literature, several process modelling languages are available [10-12]. SPEM 2.0 is simply one of them but since it has appealing features in terms of standardization, reuse, tool-support, etc. (as surveyed in [12]) as well as in terms of active community working towards its enhancement [13], it answers our expectations. SPEM 2.0 offers static as well as dynamic modelling capabilities, the latter achieved by including links to other modelling languages (e.g. UML activity diagrams). SPEM 2.0 also offers modelling capabilities to address process variability. As explored in [5] these modelling capabilities are not fully adequate to model process lines. However, the alternative modelling proposal [14], called vSPEM, which is currently matter of investigation, is still too immature to be considered in the time-frame of our project.

In SPEM 2.0, a process element (e.g. a task) can be a variability element and to it the process engineer can associate separate objects representing the differences (e.g. additions) relative to the original (called base). The variability element has an attribute that characterizes its variability type. The variability type enumeration class defines the different types of variability [8]. In Table 1, we recall those variability types used in our approach presented in Section III.

TABLE I. VARIABILITY TYPES IN SPEM2.0

Variability type	Description
na	Not applicable
contributes	Provides a way to contribute to attribute values and association instances of the base, without altering it. The base is logically replaced with an augmented variant.
replaces	Defines a replacement of a base. The replacement consists of either a complete new variant or a change concerning fundamental relationships.

SPEM2.0 is defined as a Meta Object Facility (MOF)-based meta-model, which is composed of seven main packages, which are: 1) The *Core* package defines concepts allowing for the foundation of the other packages. 2) The *Method Content*

package defines concepts allowing for the specification of a knowledge base of reusable process elements (e.g. TaskDefinition and WorkProductDefinition). 3) The *Process Structure* package defines concepts allowing for the representation of process models composed of inter-related actual process elements (e.g. TaskUse). 4) The *Process with Method* package defines concepts such as Method Content Use elements (e.g. TaskUse) for the integration of processes defined by using the concepts available in Process Structure with the instances of concepts available in Method Content. 5) The *Method Plugin* package defines mechanisms allowing for the reuse and management of method content and processes. 6) The *Process Behaviour* package defines mechanisms and concepts allowing process elements to be linked to external models for behavioural specification. 7.) The *Managed Content* package introduces concepts for managing the textual content of natural language descriptions, which are necessary to increase models understand-ability.

For a subset of the concepts (that belong to the meta-model), graphical modelling elements (icons) are at disposal. Table II recall those icons that are related to what is presented in Section III.

TABLE II. ICONS DENOTING METHOD CONTENT (USE) ELEMENTS

Task Definition	**TaskUse**
Method Content	***Method Content Use***
🟨	🟨🔄

Concerning tool-support for SPEM2.0, EPF Composer [9] is a tool that provides sufficient support with respect to our exploratory needs.

III. SAFETY- AND AUTOMOTIVE-ORIENTED PROCESS LINE

As discussed in the background section, to be able to reuse process elements, a process line approach is beneficial since commonalities and variabilities can be clearly systematized. As mentioned, currently, there is no satisfying modelling language and no tool supporting process lines. However, SPEM2.0 and EPF Composer are sufficient to implement our safety-oriented process line-based methodological framework and show its validity. Thus, in Section III.A, we illustrate how commonal-ities and variabilities can be modelled (domain engineering phase), by using SPEM2.0/EPF, and then, in Section III.B, we exploit those commonalities and variabilities to derive standard-specific single processes (process engineering phase). Fig.1 depicts the two phases and their main activities.

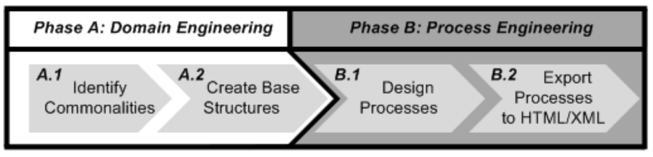

Phase A: Domain Engineering
Phase B: Process Engineering
A.1 Identify Commonalities
A.2 Create Base Structures
B.1 Design Processes
B.2 Export Processes to HTML/XML

Fig. 1. Process line phases

In the first phase we focus on the static structure of the process line. No modelling support is currently available to consider variabilities in the execution order. With respect to single processes, instead, we consider both static and dynamic structures.

A. Domain Engineering

The goal of this subsection is to explain how commonalities and variabilities are identified and modelled within SPEM2.0/EPF.

Before a process line model can be built, since all the three standards use slightly different terms to denote process elements (e.g. tasks, work products, etc.), a common terminology for all the three standards has to be defined in order to go beyond irrelevant terminological differences (Fig. 1.A.1). Development of semantic equivalence is a vital building step in process line creation. This step, which is extremely time-consuming, should be performed in close cooperation with experienced assessors to guarantee the achievement of a certifiable result. The challenge of this step is the identification and comparison of different terms within unstructured text fragments. For example, as summarized in Table III, the term *Base practice* in ASPICE is used as equivalent to the term *Activity*. Once the process elements (e.g. activities) got an identifier the mapping of different standards takes place whereby tasks are defined.

TABLE III. MAPPING OF SPECIFIC TERMS

Common identifier	**ISO 26262**	**IEC 61508**	**ASPICE**	**EPF-C**
Activity	Activity	Activity	Base practice	Task

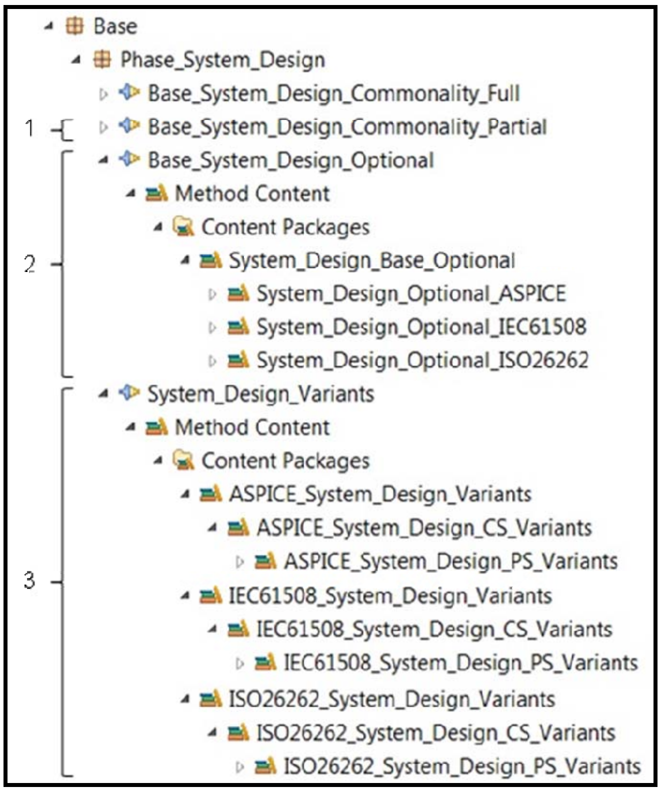
If a process element (e.g. an activity) of one standard (e.g. *System design specification* in ISO 26262) is equivalent with a process element of a different standard (e.g. *E/E/PE system design and development* in IEC 61508) the elements are mapped to a common identifier. A unique ID is given to each matching comparison in order to provide traceability to the origins of all standards and the information is collected in a spreadsheet, which allows for tracking each single element and ensuring full coverage of the standards. Additional details can be found in [18]. After having identified the commonalities and variabilities that characterize the system design phase mandated by the set of standards considered in the background, SPEM/EPF can be exploited to create a knowledge base populated by those identified commonalities and the variabilities. As recalled in Section II.C, SPEM2.0 offers a package (*Method Plugin*) that supports reuse and management of method content. Within EPF this package is implemented and thus it is possible to create plug-ins containing reusable method content. As depicted in Fig. 2, we thus decide (as it was initially proposed in [17]) to define a series of plug-ins aimed at containing base elements. We organize these plug-ins by using two logical packages (*Base* and *Processes*), which in EPF Composer are aimed at grouping method plug-ins. We use Base (respectively Processes) for organizing plugins related to the Domain (Process) engineering phase. More specifically, we define one plug-in for each type of commonality (either full or partial) and variability (i.e. optional). We also define a plug-in for all the variants that are related to either partial commonalities or variabilities.

More precisely, as Fig. 2 shows, base elements include:

Full commonality: In the defined scope (see II.A) this type of element is very rare. Therefore partial commonality is used in a flexible way.

1) Partial commonality: Elements that to be reused need to be enriched in an additive way. These elements contain a common part. For example, a new task obtained by considering the subset of steps that is in common either in all three standards or in a subset of them.

As an example of elements of type "partial commonality", we can consider the task *Define system architectural design*. This task is present in all three standards. In ISO 26262 it is called "System design specification", in ASPICE it is called *Define system architectural design* and, finally, in IEC 61508 it is called *E/E/PE system design and development*.



```
Base
  Phase_System_Design
    Base_System_Design_Commonality_Full
    Base_System_Design_Commonality_Partial
    Base_System_Design_Optional
      Method Content
        Content Packages
          System_Design_Base_Optional
            System_Design_Optional_ASPICE
            System_Design_Optional_IEC61508
            System_Design_Optional_ISO26262
    System_Design_Variants
      Method Content
        Content Packages
          ASPICE_System_Design_Variants
            ASPICE_System_Design_CS_Variants
              ASPICE_System_Design_PS_Variants
          IEC61508_System_Design_Variants
            IEC61508_System_Design_CS_Variants
              IEC61508_System_Design_PS_Variants
          ISO26262_System_Design_Variants
            ISO26262_System_Design_CS_Variants
              ISO26262_System_Design_PS_Variants
```

Fig. 2. Plug-ins for systematizing process line elements

The clauses below provide in italics the associated common steps, which justify the identification of a partial commonality:

ISO 26262: "The consistency of the preliminary architectural assumptions in ISO 26262-3:2011, 8.3.2 and the preliminary architectural assumptions in this sub-phase shall be ensured" [3]. Architectural assumptions are those related to the *representation of the structure of the item or functions or systems or elements that allows identification of building blocks, their boundaries and interfaces, and includes the allocation of functions to hardware and software elements.*

ASPICE: "Establish the system architectural design that *identifies the elements of the system* with respect to the functional and non-functional system requirements."[1]

IEC 61508: "The design shall be based on a *decomposition into subsystems with each subsystem having a specified design* and set of integration tests." [2]

2) Optional elements: Elements that might be standard-specific and that do not represent a mandatory element for each process of the process line. Optional elements can be replaced by an empty element if the single process to be derived from the process line does not include it.

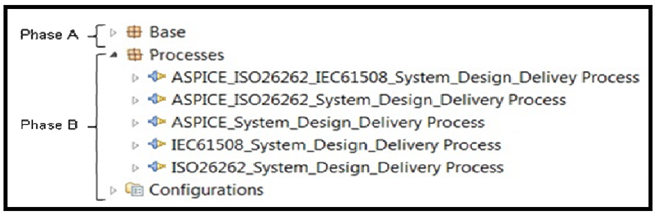
The process element *Communicate system architectural design* is optional because it is only considered in ASPICE.

3) Variant elements: Base elements also include reusable standard-specific variants. These elements are named as variants in Fig. 2 and they are obtained by enriching the elements representing partial commonalities. For the complete creation of a process line for each process element (e.g. task, work product, guideline, etc.), several variants (e.g. standard-specific, company-specific, project-specific, etc.) should be provided (Fig. 1.A.2). Typical standard-specific variants are those that deal with different safety integrity levels (SIL or ASIL). Thus, the plug-in named with the suffix *Variants* also includes process elements that are not predetermined by a standard. For the creation of company-specific (CS in Fig. 2) as well as project-specific (PS in Fig. 2) process elements, standard-specific partial commonalities should be enriched or replaced (via the *contributes* or *replaces* relationship; see Table I).

As an example of elements of type "company-specific variant elements", we can consider the task named *Define system architectural design_V_SafeCer* that replaces the base element of type "*partial commonality" named Define system architectural design* with a SafeCer specific system architecture activity. The *description* field associated to each process element is then filled in with brief information as well as pointers to the spreadsheet containing the references to the sections of the single standards.

B. Process engineering

In the previous subsection we have shown how the domain engineering phase (phase A of the process line engineering) can be performed within SPEM2.0/EPF. To proceed with phase B, we create a new plug-in for each single process that belongs to our process line and can be obtained by selecting and composing process elements contained in the base (Fig.1 B.1). Fig. 3 shows the plug-ins that should be available at the end of the process line engineering development.



```
Base
Processes
  ASPICE_ISO26262_IEC61508_System_Design_Delivey Process
  ASPICE_ISO26262_System_Design_Delivery Process
  ASPICE_System_Design_Delivery Process
  IEC61508_System_Design_Delivery Process
  ISO26262_System_Design_Delivery Process
Configurations
```

Fig. 3. The two-phase process line engineering approach

More specifically, the package "Processes" is expected to contain a plug-in for each single process of interest. Before a plug-in for a delivery process is created the area of interest has to be defined (e.g. ISO 26262). The specific process (e.g. *ISO26262_System_Design_Delivery_Process*) is built up with base tasks and the variants of the base tasks, which are provided in the *Base* package. For each delivery process a new plug-in is created where only the subsection *Delivery Process* is used. The creation of the real process occurs in the *Work Breakdown Structure* by using the tasks from the repository. To create an *Activity Diagram* related to the process the single tasks have to be properly ordered by setting their predecessors.

To create a delivery process concerning ASPICE and ISO 26262 (*ASPICE_ISO26262_System_Design_Delivery_ Process*), it is necessary to add all the needed process elements. This means that all elements in the method content *Full* and *Partial* must be added to the *Work Breakdown Structure*. Which elements from the method content *Optional* are added depends on the considered standards. If the focus is on the combination of ASPICE and ISO 26262 the elements concerning these two standards have to be added. At this point a delivery process is established which is compliant with two desired standards. Fig. 4 shows the corresponding activity diagram.

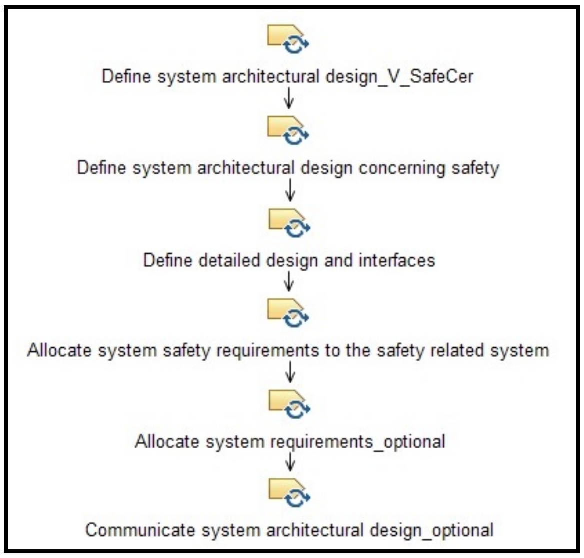

Define system architectural design_V_SafeCer
Define system architectural design concerning safety
Define detailed design and interfaces
Allocate system safety requirements to the safety related system
Allocate system requirements_optional
Communicate system architectural design_optional

Fig. 4. Activity diagram of the System design phase

Fig. 4 shows four tasks from the plug-in *Partial commonality* (upper four) whereupon the first task has been replaced by a company/consortium-specific task (this task represents the interpretation and customization carried out in the context of the SafeCer project) and two tasks from the plug-in *Optional*. The two tasks at the bottom (named *optional*) derive from the inclusion of ASPICE-specific elements.

In the EPF-Composer a *Method Configuration* is used to define a subset of the library. This subset, which is the basis for exporting to XML and HTML, defines the packages and plug-ins that are added or subtracted. EPF-Composer permits users to export libraries, plug-ins and configurations in XML format for further processing in other process-related-tools (Fig.1.B.2). Fig. 5 shows how the subset is arranged by selecting *Plug-ins* and *Content Packages*. An element appears in the configuration if the related check box is marked otherwise it is not part of the configuration.

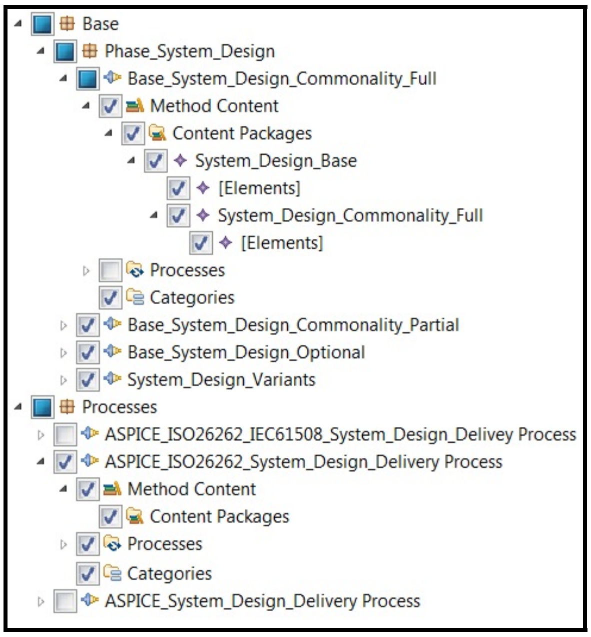

Base
Phase_System_Design
Base_System_Design_Commonality_Full
Method Content
Content Packages
System_Design_Base
[Elements]
System_Design_Commonality_Full
[Elements]
Processes
Categories
Base_System_Design_Commonality_Partial
Base_System_Design_Optional
System_Design_Variants
Processes
ASPICE_ISO26262_IEC61508_System_Design_Delivey Process
ASPICE_ISO26262_System_Design_Delivery Process
Method Content
Content Packages
Processes
Categories
ASPICE_System_Design_Delivery Process

Fig. 5. Selection of reusable process elements

The snippets below show how the task *Define system architectural design_V_SafeCer* in the variants plug-in is related the task *Define system architectural design* in the partial commonality plug-in. The linkage between single elements is implemented by using unique ID's.

```
<ContentElement xsi:type="uma:Task"
name="Define system architectural design"
id="_xJ8cgITREeOXwJvr4znqwg"
</ContentElement>
<ContentElement xsi:type="uma:Task"
name="Define system architectural design_V_SafeCer"
variabilityBasedOnElement="_xJ8cgITREeOXwJvr4znqwg"
variabilityType="replaces">...
</ContentElement>
```

IV. Lessons Learned

In this section we present the lessons learned that we have derived by implementing the safety-oriented process line approach in SPEM2.0/EPF. The lessons concern the following four main bolded aspects: **General soundness** - The adoption of the safety-oriented process line approach is sound since prescriptive processes mandated by the standards exhibit commonalities. The adoption is also beneficial since it enables systematic reuse of process elements. **Traceability** might be precondition for the acceptance of a process in a company. The clear relationship between the derived process and the original standard provides support for arguing process-compliance. Every user of the process is able to understand which section of a standard is base for the definition of a derived process element. For this reason a direct link to the standard is part of every process element. **Modelling limitations** - As pointed out by predecessors and as also found out through this work, SPEM2.0/EPF Composer offers a limited variability modelling support, which makes it not ideal for modelling a process line. **Flexible use of the notion of partial commonality** resulted to be a strategic solution for increasing the identification of common process elements.

V. Related work

The necessity to react to the introduction of the new ISO standard for functional safety has motivated several research works aimed at first of analysing the existing gap with respect to previous ways of working and then proposing solutions. In [19], for instance authors propose to simply extend ASPICE with safety-related process elements in order to fulfil the requirements of ISO 26262. This trend of extending/up-grading/amplifying a standard with safety-specific process elements in order to provide ad-hoc solutions, is also pioneered in [20-21]. Our approach profoundly differs from these ones. We do not pursue ad-hoc and "hard modelled" or (simply "hard written" in case of natural language-based proposals) solutions. Instead, we propose to thoroughly and systematically engineer a modelling framework supporting flexible process models definition and thus allowing process engineers to select and compose process elements in compliance with the required standard(s).

VI. Conclusion and future work

In this paper we have proposed a methodological frame-work for implementing the safety-oriented process line approach. More specifically, we have examined three standards that are used in the automotive domain and after having identified commonalities and variabilities we have shown how to systematically model them in SPEM2.0/EPF. Then, we have also shown how those commonalities and variabilities can be exploited for the definition of flexible processes. From this work we have drawn some lessons learned: the examined processes exhibit commonalities and thus the safety-oriented process line approach represents a sound and effective way for systematizing reuse and enabling the introduction of the changes that might be required when switching from one standard to another (e.g. for intra-domain re-certification). The current modelling support is however too limited.

In the future, in close cooperation with experienced safety assessors, we plan to extend this work by considering additional process elements and other safety related standards. The extension will include the modelling of the specific roles to synthesize the required skills to be standard compliant and ensure that a check in terms of competences has been done. Additionally, we intend to model also work products, guidance and tools. We also intend to actively contribute to the provision of a more adequate modelling support for safety-oriented process lines. Further, we aim on the definition of metrics that allow process engineers to evaluate the reduction in terms of time and cost enabled by the systematization of reuse. To be aligned with the ongoing evolution of the functional safety standards proposed by the rather influent set of automotive Swedish manufactures, we intend to consider SS-7740 [15] as well as CMMI-DEV [16] plus its corresponding extension for safety (+SAFE [20]).

Acknowledgment

This work is supported by the nSafeCer project [7].

References

[1] Automotive SPICE Process Assessment Model, Automotive SIG, 2010, www.automotivespice.com

[2] IEC61508:2010. Functional safety of electrical/electronic/programmable electronic safety-related systems.

[3] ISO26262. Road vehicles – Functional safety. International Standard, November 2011.

[4] T. Ternite. Process Lines: A Product Line Approach Designed for Process Model Development. Software Engineering and Advanced Applications, Euromicro Conference, pp. 173-180, 2009 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009.

[5] B. Gallina, I. Sljivo, O. Jaradat. Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification. Post-proceedings of the 35th IEEE Software Engineering Workshop (SEW-35), 2012.

[6] ARTEMIS-JU- 269265 pSafeCer - pSafety Certification of Software-Intensive Systems with Reusable Components.

[7] ARTEMIS-JU- 295373 nSafeCer - nSafety Certification of Software-Intensive Systems with Reusable Components.

[8] OMG. Software & systems Process Engineering Meta-model (SPEM), v 2.0. Full Specification, Object Management Group, 2008.

[9] Eclipse Process Framework, www.eclipse.org/epf/.

[10] K. Zamli and P. Lee. Taxonomy of Process Modeling Languages. Proc. ACS/IEEE Intl. Conf. Computer Sys. and Appl., Beirut, Lebanon, pp. 435–437, June 2001.

[11] S. T. Acuña, X. Ferré. Software Process Modelling. Proc. World Multiconf. Systemics, Cybernetics, and Informatics, Orlando, FL, pp. 237–242, July 2001.

[12] R. Bendraou, J.-M. Jezequel, M.-P. Gervais, and X. Blanc. A Comparison of Six UML-Based Languages for Software Process Modeling. IEEE Trans. Softw. Eng. 36, 5, pp. 662-675, September 2010.

[13] I. Ruiz-Rube, J. M. Dodero, M. Palomo-Duarte, M. Ruiz, D. Gawn. Uses and Applications of SPEM Process Models. A Systematic Mapping Study. J. Softw. Maint. Evol.: Res. Pract.; 00:1–32, Published online in Wiley InterScience. DOI: 10.1002/smr, 2012.

[14] T. Martínez-Ruiz, F. García, M. Piattini, J. Münch. Modeling Software Process Variability: An Empirical Study. IET Software, vol. 5, no. 2, pp. 172-187, 2011.

[15] SS-7740. Road vehicles - Functional Safety Process Assessment Model. SIS 2012.

[16] M. B. Chrissis, M. Konrad, S. Shrum. CMMI: Guidelines for Process Integration and Product Improvement (3rd edition) . SEI Series in Software Engineering.

[17] S. Kashiyarandi. Reusing Process Elements in Context of Safety Critical Systems Development and (re)Certification. Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, Sweden (to appear in 2014).

[18] R. Bramberger, Application of process line approach for different standards in a safety-critical context. Bachelor's thesis, Graz University of Technology, Austria (to appear in 2014).

[19] P. Johannessen, Ö. Halonen, O. Örsmark. Functional Safety Extensions to Automotive SPICE According to ISO 26262. Proceedings of the 11th International Conference Software Process Improvement and Capability Determination-SPICE. Springer, pp. 52-63. Dublin, Ireland, May 30 - June 1, 2011.

[20] +SAFE V1.2, A Safety Extension to CMMI-DEV, V1.2 (CMU/SEI-2007-TN-006). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, March, 2007.

[21] E. Petry. How to Upgrade SPICE-Compliant Processes for Functional Safety. 10th International Conference Software Process Improvement and Capability Determination-SPICE. Pisa, Italy, May 18-20, 2010.