Towards Tool-based Security-informed Safety Oriented Process Line Engineering

Inmaculada Ayala
inmaculada.ayala@mdh.se

Barbara Gallina
barbara.gallina@mdh.se

Mälardalen University
Västerås, Sweden

ABSTRACT

For the purpose of certification, manufactures of nowadays highly connected safety-critical systems are expected to engineer their systems according to well-defined engineering processes in compliance with safety and security standards. Certification is an extremely expensive and time-consuming process. Since safety and security standards exhibit a certain degree of commonality, certification-related artifacts (e.g., process models) should to some extent be reusable. To enable systematic reuse and customization of process information, in this paper we further develop security-informed safety-oriented process line engineering (i.e., engineering of sets of processes including security and safety concerns). More specifically, first we consider three tool-supported approaches for process-related commonality and variability management and we apply them to limited but meaningful portions of safety and security standards within airworthiness. Then, we discuss our findings. Finally, we draw our conclusions and sketch future work.

Keywords

Security-informed Safety; Security-informed Safety-oriented Process Lines; Tool-supported Process Customization

1. INTRODUCTION

Safety and Security represent two dependability attributes with a different focus in terms of dependability threats [3]: safety cares about avoidance of catastrophic consequences for users and the environment, while security cares about the threat of intentional unauthorized electronic interaction. Due to this different focus, two distinct communities have been developed around these two attributes. The cooperation of these communities with certification bodies has produced different normative documents (i.e. standards) aimed at guiding the development of safety and security critical systems without convergence between both domains. As a consequence, certifying a system in compliance with safety and security requirements at the same time raises many issues such as duplication of work, adoption of measures for security that conflicts with decisions taken for safety or viceversa or lack of knowledge to take appropriate decisions for safety or security just to mention a few [8].

Despite the different focus, opportunities for synergies between safety and security exist [4, 3]. In this paper, we focus on opportunities for the re-use of process-related certification artifacts. As stated, a system should be developed in compliance with the process mandated in the standard to get the corresponding certification. However, there is not one certification process that fits the development of the all possible systems. There are a lot of factors (e.g. business area, integrity level, project-specific requirements, etc.) that could cause variations in the development process and indeed, certification processes can be tailored by applying prescriptive tailoring rules. Since the tailoring is necessary, the single process should be replaced with the more realistic and applicable process line solution. A Process Line is defined as a set of similar processes within a particular domain or for a particular purpose, having common characteristics and built based upon common and reusable process assets [16]. Therefore, it is of utmost importance to have at disposal adequate modeling means to support process engineers and safety/security managers in modeling, communicating and manage flexible families of processes for the development of security informed safety critical systems.

In a previous work, we have analyzed the commonality and variability of security and safety assessment processes and sketched a possible way forward for making reuse via Security-informed Safety-oriented Process Line Engineering (SiSoPLE) [8]. A SiSoPLE is a safety-oriented process line, focused on the alignment of safety and security aspects for certification purposes. SiSoPLE is the only approach that includes a way forward for modeling security informed safety families of processes. However, up to now, SiSoPLE has been formulated as a concept and a vision for SiS processes customization. In this paper, we take one step further and explore solutions at disposal for flexible process modeling to implement SiSoPLE. Specifically, we explore the application of SPEM 2.0 [13], a solution based on Software Product Lines (SPL) [19], and the CASPER approach [1]. As case study, we use fragments of the security standard DO-326A [21] and the safety standard ARP4761 [20].

This paper is organized as follows: Section 2 provides background and related work; Section 3 explores the application of approaches for SiSoPLE; Section 4 discusses the advantages of these approaches; and we draw our conclusions and future work in Section 5.

© 2016 ACM. ISBN 123-4567-24-567/08/06. . . $15.00
DOI: 10.475/123_4

2. BACKGROUND AND RELATED WORK

In this section, we recall the essential background on which the presented work is based: security-informed safety; RTCA DO-326A; ARP4761 and variability engineering in SPEM2.0-based approaches.

2.1 SiS in airworthiness

SiS, which stands for Security-informed Safety, is an expression that has been recently introduced [4] to indicate an old truth: "For a system to be safe, it also has to be secure". In some circumstances, the threats that hinder dependability can be equivalent for safety as well as security. Within the airworthiness state of practice, safety and security are treated separately and appropriate standards, namely ARP4761 [20] and DO-326A/ED-202A [21], including different assessment processes, have been proposed.

In recent years, however, the scientific community working within industrial projects has started proposing the alignment of the two processes [17, 8] but such alignment is still in its infancy. In this paper, we continue the work aimed at aligning RTCA DO-326A/ED-202A and ARP4761. Thus, we here recall essential information on these two standards and more specifically, on some of their sub-processes. DO-326A provides guidance to handle the threat of intentional unauthorized electronic interaction to aircraft safety. It is composed of several activities and tasks but in this paper we focus on the Preliminary Aircraft Security Risk Assessment (PASRA) and on the Preliminary System Security Risk Assessment (PSSRA). The goal of PASRA plus PSSRA is to identify threat conditions and threat scenarios and assessing all security risks at the aircraft and system level. ARP4761 includes two similar processes, called Aircraft-Level and System-Level Functional Hazard Assessment (FHA). AFHA and SFHA are focused on safety. Thus, they are aimed at identifying failure conditions and assessing all safety risks at the aircraft and system level.

PASRA takes as input three work products: the architecture under consideration, failure conditions and severity, and the information related to the security environment and perimeter. Based on these inputs, the following tasks are performed: threat condition identification and evaluation; threat scenario identification; security Measure Characterization; and level of threat evaluation. The final outcome of PASRA is the preliminary security effectiveness objectives based on identified and evaluated threat conditions. **AFHA** takes as input the list of top-level functions plus the initial design decisions of the aircraft. Additionally, it considers as input the aircraft objectives and requirements. Based on these elements, the following set of steps is performed: identification of all functions and corresponding failure conditions; determination of effects of the failure conditions and classification of the determined effects. The final outcome of AFHA is the safety objectives and the derived safety requirements based on identified and evaluated failure conditions. Process at the system level (PSSRA and SFHA) have the same structure but target aircraft systems, so they have minor differences with regard to inputs, outcomes and focus on some tasks.

2.2 SPEM 2.0-based variability engineering

Various means are at disposal to model processes [11]. In this paper, we select SPEM 2.0 [13], which is the Object Management Group (OMG) standard for process modeling. SPEM 2.0 offers support for systematic reuse, which is crucial for dealing with SiS. Moreover, SPEM 2.0 is tool supported by means of the EPF composer [7] and the Rational Method Composer [5]. In this paper, we use the EPF composer, which is open source.

The conceptual framework of SPEM 2.0 considers two views, the Method content and the Process packages. The goal of the Method Content package is to set up a knowledge base of intellectual capital for software development that would allow them to manage and deploy their content using a standardized format. Elements that are defined in this package are tasks, roles, tools and support material like white papers, principles or best practices. In contrast, the Process package focus on supporting the systematic development, management, and growth of development processes. This is the package in which development process are actually defined using elements that point to elements of the Method Content package. In order to define a process, tasks are organized in activities, phases and iterations. Additionally, SPEM 2.0 defines an structure known as process pattern (capability pattern in the EPF composer) that represents a reference process for a specific discipline and technology that can be used for quickly assembling processes based on project needs.

SPEM 2.0 has support for variability management in the Method Content package. Specifically, it offers a mechanism based on pairs of elements known as *base* and *extension*. So, it is possible to define an *extension* by means of its relationship with a *base*. SPEM 2.0 defines 5 types of variability relationship *na* (by default), *contributes*, *replaces*, *extends* and *extends and replaces*. In this work we apply *contributes* and *extends*. The *contributes* variability allows the definition of an *extension* from a *base* in an additive fashion without altering any of the existing properties of the *base*. The behavior of the *extends* variability is similar but allows to override properties of the *base* element.

Several works have highlighted limitations of SPEM 2.0 to manage variability in process lines [14, 18, 2]. The main limitation of this standard is the variability support is intended to make faster the definition of new process elements and does not provide guidance for the reuse of process elements between similar processes. So, by instance, it is not possible to model issues like in which context a process element can be replaced by another or the order of the tasks modified. In typical process line approaches, we have a reference process that represents a family of processes to develop systems in a particular domain. This reference process contains the necessary information to tailor itself to meet the requirements of a specific project. As SPEM 2.0 does not provide guidance for the reuse of the process elements, this reference process cannot be defined. Therefore, the derivation of a new process for a specific project entails the selection of elements of the Method Content package one by one selecting variants when it is required taking into account information provided in the standards. In the domain of the SiS process, with process elements that comes from different standards the composition of new processes is even more difficult due to the number of process elements that can be included in the process and the variability relations between them. Additionally, the certification requirements of this domain makes even more important to not forget the inclusion or exclusion of specific process elements in specific development contexts.

In order to overcome this limitation, several works have

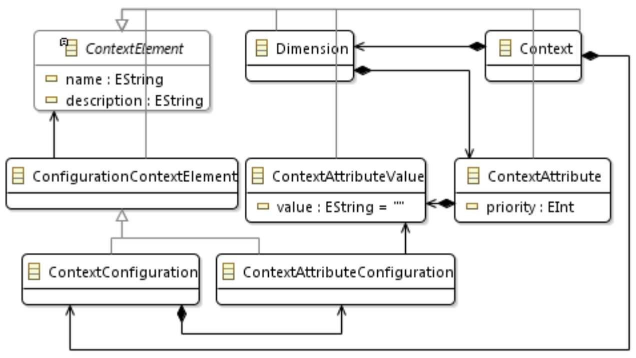


Figure 1: Context metamodel of CASPER.

extended SPEM 2.0 with mechanisms of the SPL [19]. These extensions have been performed by integrating process models with SPL tools [14] or extending process metamodels with variability mechanisms of SPLs [18]. The problem of existing solutions is they are difficult to be applied in real scenarios. Some proposals are based on tools that are currently outdated [2] or are not at disposal [18]. Additionally, the integration of existing SPL tools like the BVR tool [12] with process tools like EPF is difficult to achieve.

The CASPER approach [1] intends to overcome the limitations of SPEM 2.0 to model reference process, but in a manner different to the mentioned works. The rational behind these proposals is to model all possible alternatives in the same process and select one of them. In contrast, CASPER proposes to have a process that can be transformed to meet the requirements of a specific project. This proposal uses two models as input, a software process defined in a simplified variant of SPEM 2.0 (eSPEM) and a model of the organizational context of the enterprize for a specific project. Then, using a set of ATL transformation rules [15] a new adapted process model is generated. The key idea of the authors is to codify the knowledge of the process engineer to tailor process according to the context in the ATL transformation rules. This approach has been validated in the context of two small enterprizes. The main advantage of this proposal is it relies on ready to use technologies, so it is relatively easy to re-use results provided by this work. On the other hand, the definition of the transformation rules could be very complex even for simple processes and context models.

The eSPEM metamodel has the same structure as SPEM 2.0, so it distinguishes between the Method Content and the Process package. However, it does not have *Guidance* elements like white papers, methods or tool mentors. This is a great disadvantage to model SiSoPLE because methods and tools are an important part in certification processes. On the other hand, the main novelty of CASPER is its context metamodel (see Figure 1). The root class of this metamodel is *Context* that is composed of *Dimension* and *ContextConfiguration* elements. In *Dimension*, we model all attributes of the context and their possible values using *ContextAttribute* and *ContextAttributeValue*. In *ContextConfiguration*, we select a specific configuration of the context using *ContextConfigurationAttribute* elements. The work of the ATL transformation rules is to analyze the reference process modeled in eSPEM and to modify its process elements taking into account values of *ContextConfiguration*.

3. TOWARDS TOOL-SUPPORTED SISOPLE

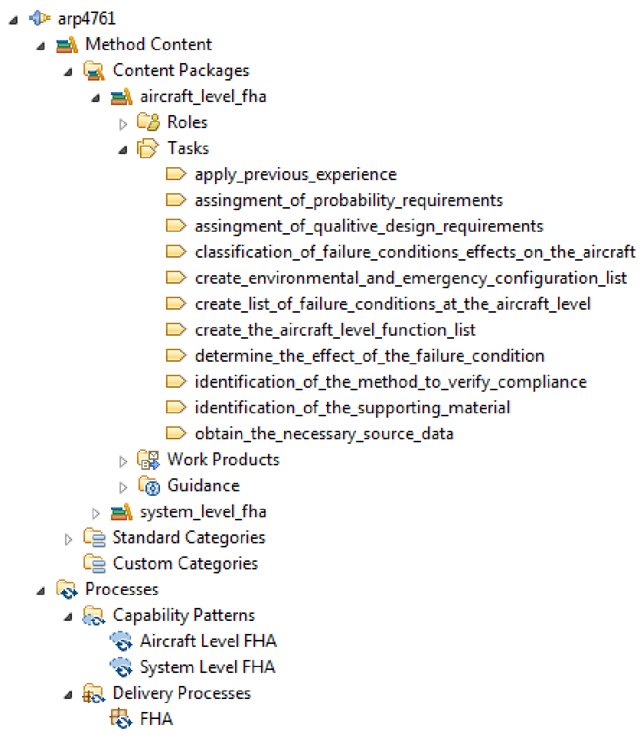
Prior to the application of the different proposals for SiSoPLE, it is necessary to analyze the two proposed standards to find the commonalities and the variabilities of the process fragments. In order to derive the commonalities of these processes we use two tools. Firstly, in order to increase the commonalities between the two processes, we use the definitions of *partial commonality* and *full commonality* proposed in [10]. *Partial commonality* is applied whenever process elements of the same type expose at least one common aspect, while *full commonality* is applied whenever process elements of the same type expose only common aspects.

Secondly, we consider the terminological framework for dependability and security proposed in [3]. We will make some assumptions about this common framework and some simplifications, since the focus of this paper is the exploration of the use of tools for SiSoPLE. By instance, we are going to assume that all the tasks are performed by the analyst, and to derive the commonality we will focus on input and outputs of the tasks. PASRA and AFHA (and attached processes for the system level PSSRA and SFHA) are processes that focus on the identification of failure conditions and assessing the risk at the corresponding level (aircraft or system level). Taking into account the mentioned framework, we assume that failure conditions might include conditions coming from malicious and non malicious behaviors and thus a partial commonality could be identified. Additionally, we assume that risk is the same as the possibility of failure. Taking into account these simplification, we find five categories of partially and full common tasks in these processes: (i) gathering of information for identification and classification of failure conditions; (ii) identification of failure conditions; (iii) classification of severity of failure condition effects; (iv) assignment of probability to failures; and (v) update of requirements and architecture at the corresponding level. In addition, there are tasks that are specific to the standards like the information to the development groups or to find documentation and verification methods.

3.1 The EPF composer

In this section, we present the modeling of the SiS process line using SPEM 2.0 and the EPF composer. As stated, one of the strongest point of SPEM 2.0 is the concept of library of process elements also known as the Method Content package. In order to define the SiSoPLE, we need to define process elements to compose processes contained in both standards, and additionally, the process elements that enable the modeling of a process aligned with the security and safety standards. This last step is the most difficult because it requires to find the full-commonality, partial-commonality and the variability at the level of tasks for both standards. In order to model the partial commonality, we use the variability mechanisms of SPEM.

The modeling of the process elements and their relative order is derived from the analysis of the corresponding standard. In EPF, these contents are encapsulated in an structure known as Method plug-in. In Figure 2, we can see the Method plug-in for the FHA process. In this Figure, we can distinguish the Method Content and the Process packages. In the first one, we can see tasks defined for the AFHA and SFHA and in the second one, capability patterns for both levels and a delivery process for the FHA. Tasks are used to compose the two *Capability Patterns*. The process asso-



arp4761
- Method Content
  - Content Packages
    - aircraft_level_fha
      - Roles
      - Tasks
        - apply_previous_experience
        - assingment_of_probability_requirements
        - assingment_of_qualitive_design_requirements
        - classification_of_failure_conditions_effects_on_the_aircraft
        - create_environmental_and_emergency_configuration_list
        - create_list_of_failure_conditions_at_the_aircraft_level
        - create_the_aircraft_level_function_list
        - determine_the_effect_of_the_failure_condition
        - identification_of_the_method_to_verify_compliance
        - identification_of_the_supporting_material
        - obtain_the_necessary_source_data
      - Work Products
      - Guidance
    - system_level_fha
  - Standard Categories
  - Custom Categories
- Processes
  - Capability Patterns
    - Aircraft Level FHA
    - System Level FHA
  - Delivery Processes
    - FHA

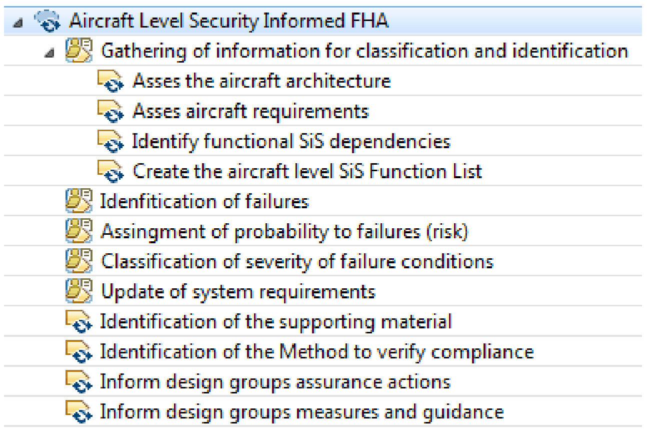
Figure 2: EPF plugin for ARP4761.

ciated to the standard is obtained copying these capability patterns in a *Delivery Process* and setting that the SFHA is repeated for each system of the aircraft. We follow the same procedure to model PASRA and PSSRA of DO-326A and the SiS process that aligns both standards. To derive tasks of this last process, we analyze the commonality relations between tasks of both standards taking into account the framework for dependability and security. We analyze the structure of the partial variability and apply variability extension that allows a higher re-use of the attributes and relations of the tasks.

In order to illustrate the derivation of the SiSoPLE tasks, we focus on tasks devoted to the gathering of information for identification and classification of failure conditions. In this case, we have two couples of tasks that have partial-commonality, *Identify functional security dependencies* and *Create the Environment and Emergency Configuration List*, and *Identify Security Measures* and *Create the aircraft level Function List*.

The partial commonality of *Identify functional security dependencies* and *Create the Environment and Emergency Configuration List* can be modeled using a *contributes* variability. The first task identifies the functional security dependencies but it does not consider environmental and emergency factors, which are important in the safety domain. The *contributes* variability allows to extends a base tasks in an additive way without directly altering any of the existing properties of this task. So, we create a new task *Identify functional SiS dependencies* that has a relation of contribution with *Identify functional security dependencies* and add a new element to the output of the task, the Environment and Emergency Configuration List.

The tasks *Identify Security Measures* and *Create the aircraft level Function List* have a relation of partial commonality that can be modeled using *extends*. Both tasks have the same goal, but the input of the first one is the aircraft architecture and requirements, while the input of the second one is source data. On the other hand, outputs of these tasks are the security measures list and the external and internal functionality at the aircraft level, respectively. Taking into account the level of detail provided in these tasks and the mentioned framework, one task that align the Safety and Security with the same goal as them should have as the input the architecture and requirements of the aircraft (from PASRA) and as the output the list of external and internal functions (from AFHA). The *extension* variability allows to use all the attributes of the base element (i.e. *Identify Security Measures* or *Create the aircraft level Function List*) and at the same time, to override inherited properties with the values of our election. In our case, we create a new task *Create the aircraft level SiS Function List* that extends *Create the aircraft level Function List* and overrides the input with the value Aircraft architecture and requirements.



- Aircraft Level Security Informed FHA
  - Gathering of information for classification and identification
    - Asses the aircraft architecture
    - Asses aircraft requirements
    - Identify functional SiS dependencies
    - Create the aircraft level SiS Function List
  - Idenfitication of failures
  - Assingment of probability to failures (risk)
  - Classification of severity of failure conditions
  - Update of system requirements
  - Identification of the supporting material
  - Identification of the Method to verify compliance
  - Inform design groups assurance actions
  - Inform design groups measures and guidance

Figure 3: Capability Pattern for the SiS process.

After the analysis of commonality and variability and the derivation of the new tasks, we can model the capability pattern for the Aircraft-Level (see Figure 3). This pattern contains activities that model the four type of tasks with full and partial commonality. Additionally, tasks that are not common between processes are added to the capability pattern as they are taking into account their relative order with other tasks of the process. In this case, tasks related to the verification of safety cases and information of the development groups (see bottom of Figure 3) are placed in the end of the capability pattern. We repeat the same process for the System-Level to get its capability pattern, and with these two capability patterns, we model the SiS process that aligns AFHA, SFHA, PASRA and PSSRA.

3.2 Process model extended with SPL concepts

As stated in Subsection 2.2, there are not solutions based on SPL tools that can be used in real scenarios. So, we have implemented our own extension of SPEM 2.0 with SPL concepts for testing purposes. This extension is implemented using as a base the UMA metamodel [6], which is one of the main components of the EPF composer. Extended UMA (see Figure 4) focus on the Process package at two levels, process elements (i.e. Role Descriptor, Work Product Descriptor and Task Descriptor) and process structures (i.e. Capability Pattern, Activity, Phase and Iteration). Our goal is to make possible the definition of reference processes. So, we create a new abstract class named *VariationPoint* that extends *BreakDownElement*, which is the super class of all elements that can be placed in a process. With this extension, we can model a process that is composed of elements from the original SPEM and elements that extend *VariationPoint*. This class models a part of the process that can vary and has specializations for each type of SPEM 2.0 element that can vary in a process. We illustrate the work of this class focusing on one of its specializations, *TaskVP*. If our reference process has more than one task that can be placed in the same part of the process, we add an instance of *TaskVP*. This class has a reference to all the *TaskDescriptor* that can be placed in this part of the process named *occupation*. The fields *min* and *max* (inherited of *VariationPoint*) indicate the minimum and the maximum number of *TaskDescriptor* elements from *occupation* that can be placed in this part of the process. *TaskVP* extends *WorkBreakDownElement* too, which is the class that extends all the process elements that have an ordering in the process (i.e. *Activity*, *CapabilityPattern*, *TaskDescriptor*,...). Process elements that do not require ordering (i.e. *RoleDescriptor* and *WorkProductDescriptor*) do not extend *WorkBreakDownElement*. We apply the same mechanism to model variation points for process structures, i.e. *Phase*, *CapabilityPattern*, *Iteration* and *Activity*.

In order to model the SiSoPLE, we firstly model the tasks that can be part of the process using the same procedure that we have followed for SPEM 2.0 in Subsection 3.1. So, we model tasks corresponding to both standards and derive the SiS-informed tasks using the variability support of SPEM. Then, we model the SiSoPLE using our extension.

The resultant SiSoPLE (see Figure 5) contains elements from the original SPEM 2.0 and our extension. On the one hand, we model common process elements of the process line using original SPEM 2.0 elements, so the process is composed of two capability patterns, one for the Aircraft-level and another for the System-level. On the other hand, we use specializations of *VariationPoint* to model alternatives for a process element or optional elements. We have three exclusive alternatives to model the task identification of dependencies depending on if the process is PASRA, AFHA or SiS process, so they are included in the *occupation* relation of the *identify_dependencies_vp* with a *min* and *max* set as 1. Optional elements like *identification_of_the_supporting_material* are modeled using *TaskVP* with a *min* set as 0 and a *max* set as 1.

The work of this metamodel requires additional tooling which is out of the scope of this contribution. Firstly, it needs a tool that allows to select and generate variants of our reference process taking into account the cardinality given in the *VariationPoint*. Secondly, it needs constructs to model and enforce cross-tree constraints. These constraints are a common tool in the SPL domain and determine the type of relation between arbitrary process elements using logic formulae. On the other hand, there are other elements of SPEM 2.0 that has not been included in Figure 4 but they are part of our extension, like *Guidance* and *Tool*.

3.3 Process transformation using CASPER

In order to apply CASPER for the SiSoPLE, we proceed as in the case of the extension of SPEM 2.0 but the modeling of the reference process will be different. The important aspect for the CASPER approach is to have a process modeled in eSPEM that can be transformed, so the reference process can be any of the process that can be modeled in our process line. For this example, we choose the process depicted in Figure 3. In order to keep the example as simple as possible, the model of the context only considers the domain of the process that can be security, safety or SiS (see Figure 6). Additionally, the context is configured for the security.

Guidelines for the definition of the transformation process are given in [1]. An ATL transformation process consist of a set of transformation rules and functions that facilitate the transformation process known as helpers. CASPER has two types of transformation rules, one for commonalities and another for variabilities. Rules for commonalities focus on generate a copy of a process element from the input process model. Rules for variabilities vary the process elements according to the context configuration. The detection of the process elements that vary and the assignment of new values is implemented using helpers. The rule for variable tasks is depicted in Figure 7. To detect that the *Task* can vary depending on the context is used the helper *optionalRule* (line 2), which internally contains a list of all the tasks that can vary in the reference process. Then, the body of the rule (lines 8-13) makes an exact copy of the incoming element but the *task* (line 10). The new value of the *task* is given by the helper *selectTaskRule* that analyzes the context and returns the appropriate task. The result of the transformation process is the PASRA and PSSRA processes from DO-326A.

4. DISCUSSION

As we stated in the introduction, our focus is to explore the potential of SPEM 2.0, extended UMA and CASPER to implement SiSoPLE. In this discussion, we focus on aspects relevant for the definition of the reference process in the SiS domain. Specifically, aspects under discussion are semantic richness, extensibility and automation.

The semantic richness of proposals' metamodels (UMA from EPF, extended UMA and eSPEM from CASPER) is very similar because they are based in the SPEM 2.0 metamodel. However, extended UMA has constructs to define a reference process with variant process elements, which is not possible in UMA and eSPEM. Additionally, eSPEM has the disadvantage of not having support for *Guidance* process elements. An important limitation of the three metamodels is they do not have concepts for the security and safety domain that are important for certification purposes. Specifically, it is not possible to model criticality levels and certification requirements. Moreover, it is not possible to generate a log of the execution of the certification process. In a previous work, we explored the semantic extension of safety process lines [9] using criticality levels and stereotypes.

The extensibility of the proposals is another important factor because standards, tools and methods evolve quickly in the security and safety domains. In the appearance of new process elements, the three proposals require the modeling of new elements in the Method Content package using the variability mechanisms of SPEM. However, this affects in a different way to the already modeled processes. In the

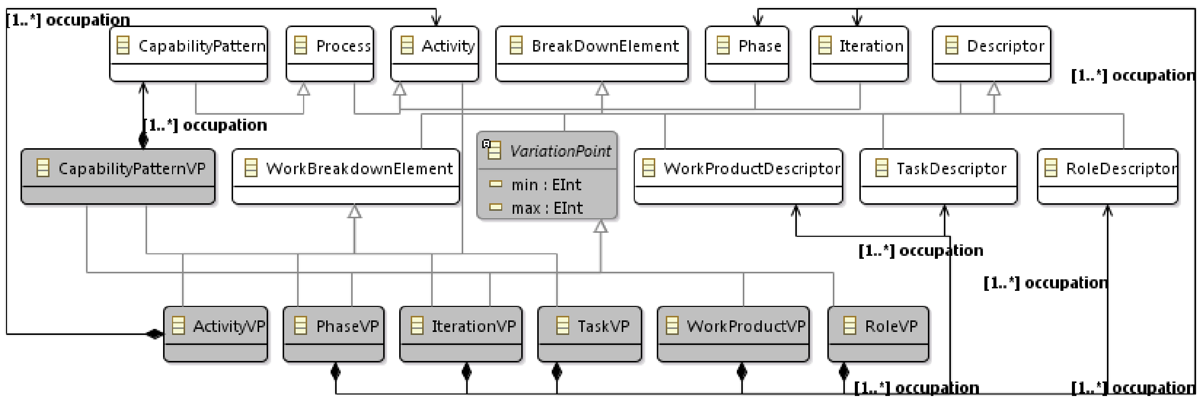
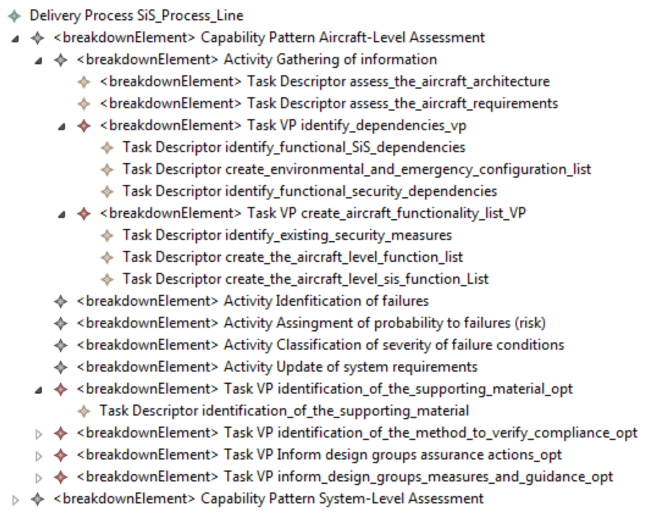
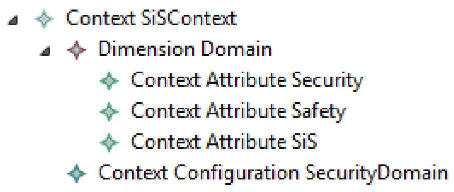


Figure 4: Partial view of the extended version of the UMA metamodel.



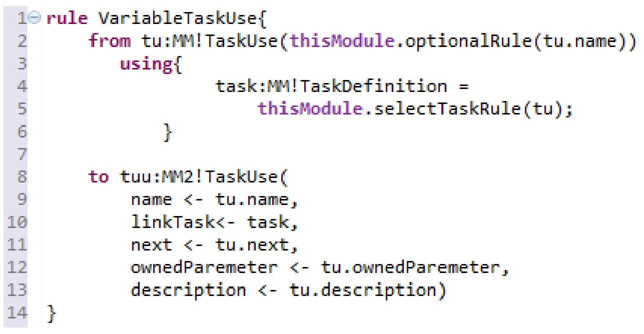
Delivery Process SiS_Process_Line
<breakdownElement> Capability Pattern Aircraft-Level Assessment
<breakdownElement> Activity Gathering of information
<breakdownElement> Task Descriptor assess_the_aircraft_architecture
<breakdownElement> Task Descriptor assess_the_aircraft_requirements
<breakdownElement> Task VP identify_dependencies_vp
Task Descriptor identify_functional_SiS_dependencies
Task Descriptor create_environmental_and_emergency_configuration_list
Task Descriptor identify_functional_security_dependencies
<breakdownElement> Task VP create_aircraft_functionality_list_VP
Task Descriptor identify_existing_security_measures
Task Descriptor create_the_aircraft_level_function_list
Task Descriptor create_the_aircraft_level_sis_function_List
<breakdownElement> Activity Idenfitication of failures
<breakdownElement> Activity Assingment of probability to failures (risk)
<breakdownElement> Activity Classification of severity of failure conditions
<breakdownElement> Activity Update of system requirements
<breakdownElement> Task VP identification_of_the_supporting_material_opt
Task Descriptor identification_of_the_supporting_material
<breakdownElement> Task VP identification_of_the_method_to_verify_compliance_opt
<breakdownElement> Task VP Inform design groups assurance actions_opt
<breakdownElement> Task VP inform_design_groups_measures_and_guidance_opt
<breakdownElement> Capability Pattern System-Level Assessment

Figure 5: SiSoPLe modeled in extended UMA.



Context SiSContext
Dimension Domain
Context Attribute Security
Context Attribute Safety
Context Attribute SiS
Context Configuration SecurityDomain

Figure 6: Context model configured for the Security.



```
rule VariableTaskUse{
    from tu:MM!TaskUse(thisModule.optionalRule(tu.name))
       using{
                task:MM!TaskDefinition =
                    thisModule.selectTaskRule(tu);
            }

    to tuu:MM2!TaskUse(
        name <- tu.name,
        linkTask<- task,
        next <- tu.next,
        ownedParemeter <- tu.ownedParemeter,
        description <- tu.description)
}
```

Figure 7: ATL Transformation rule for variable tasks.

case of EPF, we should analyze if the new process element affects to an already defined process. In this case, each affected process should be modified accordingly. In the case of extended UMA, only the reference process should be modified. Finally, CASPER does not require any modification of the reference process but the modification of the transformation rules. In our opinion, the proposal with the most challenging extension mechanism is EPF because it requires the inspection of each defined process and their modification. On the other hand, the extension of the CASPER approach can be challenging too because if the new process element is affected by the context, it requires to modify the transformation process. In this case, in the simplest situation, it is necessary to add a new helper and modify rules to detect variable process elements and the assignment of the appropriate process element according to the context. The risk in the extension of the CASPER is to make a modification in the rules that affects an already existing rule. So, additional testing will be required to check that the transformation process works properly. Therefore, we consider extended UMA is the easiest to be extended.

Finally, automation is an important feature for SiSoPLE because it facilitates the work of process engineers and avoids to forget the inclusion in the process important elements for certification. In this regard, only CASPER supports the automatic generation of processes.

Table 1 summarizes the results of this discussion. For each discussed aspect, the proposal that achieves the best score is marked with +++, the second with ++ and the worst with +. According to this, there is not a perfect solution to implement SiSoPLE but the most promising solution for SiSoPLE is extended UMA. Its only limitation is with regard to the automation, a feature that is only present in the CASPER approach. In our opinion, an optimal solution to implement SiSoPLE would be an integration of the EPF composer with a variability management tool like BVR [12]. This solution provides the semantic richness and extensibility of extended UMA and the automation of the CASPER approach. However, the integration of these tools implies to overcome technical limitations that are out of the scope of this exploratory study.

5. CONCLUSION AND FUTURE WORK

To reduce the cost and the time needed for process-related certification within security-informed safety-critical systems, systematic reuse represents a key solution and could be pur-

Table 1: Summary of the discussion

	Semantic richness	Extensibility	Automation
EPF	++	+	+
Extended UMA	+++	+++	+
CASPER	+	++	+++

sued via SiSoPLE. In this paper, we have conducted a study aimed at comparing available off the shelf tool-supported modeling capabilities, which could serve the purpose of engineering SiSoPLEs. To perform our comparison we have defined a set of criteria and selected limited but generic enough portions of safety and security standards, in use within airworthiness, namely DO-326A and ARP4761. Specifically, we have used the EPF composer, a process model extended with concepts of the SPL domain and the CASPER approach. From our study, it emerged that the most promising approach for SiSoPLE is an integration of the EPF composer with a tool for variability management like BVR.

As future work, we plan to further develop SiSoPLE in various directions. Firstly, we plan to define a domain and application engineering process that takes into account the requirements of the SiS domain and the convergence between certification standards. Additionally, as the most promising approach for SiSoPLE is the integration with SPLs by means of variability management tools, we plan to use available metrics of this domain to define metrics that allow to evaluate the reduction in terms of time and cost enabled by the systematization of reuse.

6. ACKNOWLEDGMENT

This work is supported by EU and VINNOVA via the ECSEL Joint Undertaking under grant agreement No 692474, project name AMASS.

7. REFERENCES

[1] J. A. H. Alegría and M. C. Bastarrica. Building software process lines with casper. In *International Conference on Software and System Process*, 2012.

[2] F. A. Aleixo, M. A. Freire, W. C. dos Santos, and U. Kulesza. Automating the variability management, customization and deployment of software processes: A model-driven approach. In *12th International Conference on Enterprise Information Systems. Selected papers.* Springer, 2011.

[3] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[4] R. Bloomfield, K. Netkachova, and R. Stroud. Security-informed safety: If it's not secure, it's not safe. In *Proc. of 5th International Workshop on Software Engineering for Resilient Systems (SERENE).* Springer, 2013.

[5] I. Corporation. Rational method composer. http://www-03.ibm.com/software/products/es/rmc. Accessed: 2016-08-30.

[6] E. Foundation. Eclipse process framework composer 1.0: Architecture overview. http://www.eclipse.org/epf/composer_architecture/. Accessed: 2016-08-29.

[7] E. Foundation. Eclipse process framework project (epf). https://eclipse.org/epf/. Accessed: 2016-07-12.

[8] B. Gallina and L. Fabre. Benefits of security-informed safety-oriented process line engineering. In *IEEE/AIAA 34th Digital Avionics Systems Conference*, 2015.

[9] B. Gallina, K. R. Pitchai, and K. Lundqvist. S-TunExSPEM: Towards an extension of SPEM 2.0 to Model and Exchange Tunable Safety-Oriented Processes. In *Software Engineering Research, Management and Applications*. Springer, 2014.

[10] B. Gallina, I. Sljivo, and O. Jaradat. Towards a safety-oriented process line for enabling reuse in safety critical systems development and certification. In *35th IEEE Software Engineering Workshop*, 2012.

[11] L. García-Borgoñón, M. A. Barcelona, J. A. García-García, M. Alba, and M. J. Escalona. Software process modeling languages: A systematic literature review. *Information and Software Technology*, 56(2):103 – 116, 2014.

[12] M. R. Group. BVR tool. http://modelbased.net/tools/bvr-tool/. Accessed: 2016-07-27.

[13] O. M. Group. Software & systems process engineering meta-model specification. Technical Report formal/2008-04-01, Object Management Group, 2008.

[14] J. A. Hurtado Alegría, M. C. Bastarrica, A. Quispe, and S. F. Ochoa. An mde approach to software process tailoring. In *Proc. of the 2011 International Conference on Software and Systems Process*, 2011.

[15] F. Jouault and I. Kurtev. Transforming models with atl. In *Proc. of the 2005 International Conference on Satellite Events at the MoDELS*. Springer, 2006.

[16] M. Kuhrmann, D. M. Fernández, and T. Ternité. Realizing software process lines: Insights and experiences. In *Proc. of the International Conference on Software and System Process*, 2014.

[17] S. Lautieri, D. Cooper, and D. Jackson. Safsec: Commonalities between safety and security assurance. In *Proc. of the 13th Safety-critical Systems Symposium*. Springer, 2005.

[18] T. Martínez-Ruiz, F. García, and M. Piattini. Towards a SPEM v2.0 extension to define process lines variability mechanisms. In *Software Engineering Research, Management and Applications*. Springer, 2008.

[19] K. Pohl, G. Bückle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 1st edition, 2010.

[20] S. S-18. ARP4761: Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. Technical report, SAE International, 1996.

[21] SC-216. DO-326: Airworthiness security process specification. Technical report, Radio Technical Commission for Aeronautics, 2010.