An integrated approach to support process-based certification of variant-intensive systems

Lucas Bressan[1], André L. de Oliveira[1], Fernanda C. Campos[1], Yiannis Papadopoulos[2] and David Parker[2]

[1] Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora MG, Brazil
{lucasbressan, andre.oliveira, fernanda.campos}@ice.ufjf.br
[2] University of Hull, U.K.
{d.j.parker, y.i.papadopoulos}@hull.ac.uk

Abstract. Component-based approaches and software product lines have been adopted by industry to manage the diversity of configurations on safety-critical software. Safety certification demands compliance with standards. ISO 26262 standard uses the concept of Automotive Safety Integrity Level (ASIL) to allocate safety requirements to components of a system under design. Compliance with standards is demonstrated through achieving those ASILs which can be very expensive when requirements are high. While achieving safety certification of variant-intensive components without being unnecessarily stringent or expensive is desirable for economy, it poses challenges to safety engineering. In this paper, we propose an approach to manage the diversity of safety goals and supporting safety certification of software components. Our approach is built upon the integration among ASIL decomposition, software process modeling, and variability management techniques. The approach supports cost-effective safety certification and the efficient tailoring of process models to components according to their ASILs. We evaluated our approach in the automotive domain. The approach is feasible in supporting the management of the diversity of safety goals, and cost-effective safety certification of software components.

Keywords: Safety certification, Safety critical software, Software development process, Model-based engineering.

1 Introduction

Safety-critical systems are systems in which failures may lead to catastrophic consequences to the environment and/or to people involved with their operation. This critical nature demands addressing dependability properties, e.g., safety, reliability. Safety standards provide guidance to analyze and demonstrate safety properties at different levels of abstraction. The ISO 26262 [10] automotive standard prescribes a set of safety goals to be achieved, activities to be performed, and artefacts to be produced, depending on the criticality of an item, stated through an Automotive Safety Integrity Level (ASIL). The development lifecycles of automotive system components (items) may change according to their assigned ASILs. Safety goals and ASILs are assigned to a function via systematic evaluation of severity, probability of occurrence and controllability of a hazardous event. Highly critical functions demand addressing more expensive safety goals and development lifecycle processes in comparison with functions that pose lower risks to the overall safety. Assigning stringent ASILs to classify the risk of failures on less critical system functions may incur in unnecessary certification costs [15, 16]. To counter this, prescriptive safety standards [10, 23] establish rules for decomposing ASILs assigned to top-level failure conditions (hazards) through contributing component faults.

Recent extensions in the scope of ISO 26262 have included support for functional safety on all road vehicles, with the introduction of requirements on trucks, buses, trailers, semi-trailers, motorcycles and their supporting processes. Such extensions introduce more variability in the development of automotive systems. Component-based approaches and Software Product Lines (SPL) have been adopted in the automotive [24] and aerospace [7] industry for their benefits of reduction of the time to market and development effort, and increased product quality [21]. SPL approaches have been extended to consider safety engineering and certification issues [15, 24]. A SPL is a variant-intensive architecture with common and variable functions shared among different systems from an application domain. Common and variable functions can be combined to derive different configurations. In variant-intensive automotive systems, variation in the design choices and usage context may impact hazard analysis, assignment of top-level safety goals (ASILs) and their decomposition through components [7, 15, 16].

Existing Model-Based Safety Assessment (MBSA) techniques provide automated support for ASIL decomposition in standalone [1, 19, 20] and variant-intensive system architectures [16]. ASIL decomposition results provide information to support the management of the diversity of safety goals and cost-effective safety-certification of system and software components in compliance with safety standards. Lifecycle models defined in cross-domain standards can be specified with the support of OMG Software & Systems Process Engineering Metamodel (SPEM) version 2.0 [17] compliant modeling tools, e.g., EPF Composer[1]. The integration of process modeling and variability management techniques [8] within AMASS[2] Platform enables variant management on EPF software process models. However, achieving safety certification and deriving EPF process models for variant-intensive software components without being unnecessarily stringent is challenging. Moreover, the manual configuration of EPF process models for each software intensive component in complex and large-scale system architectures can be burden. In addition, changes in the system design may impact on the ASIL allocation at the system level and decomposition at the component level, leading to modifications on the safety goals, and consequently the reconfiguration/generation of EPF process models for each individual component.

This paper proposes an approach, enhancing Oliveira et al. [16] work, supporting the management of the diversity of safety goals and cost-effective safety certification of variant-intensive components. It integrates MBSA and ASIL decomposition techniques, EPF Composer and BVR tools within the AMASS Platform. We evaluated

[1] https://www.eclipse.org/epf/
[2] https://www.amass-ecsel.eu/content/about

our approach in an automotive braking system. This paper is organized as follows: **Section 2** presents the background information needed for the reader understanding our approach. **Section 3** presents our approach to support process-based certification of variant-intensive software components and its evaluation in the automotive domain. In **Section 4,** we discuss the related work. Finally, **Section 5** presents the conclusions and future work.

2 Background

2.1 ISO 26262

ISO 26262 is a safety standard that postulates requirements for functional safety on electrical and electronic systems embedded into small and medium sized (up to 3.5 tons) general purpose road vehicles [10]. It is important for the development of software intensive systems.

ASILs are initially assigned to classify the risks that hazards pose to the overall safety, during the ISO 26262 Part 3 – Concept phase [10], after 3-7 Hazard Analysis and Risk Assessment. A hazard is a "*potential source of harm caused by malfunctioning behavior of the item*". ASILs are assigned based on the *severity* of the harm, the *probability* of exposure to operational situations, and *controllability* of each hazardous event at the 3-8: Functional safety concept. A safety goal is then derived for each hazard, according to its ASIL. Safety goals are top-level safety requirements, from which functional safety requirements are derived, thus, characterizing the Functional Safety Concept. The definition of the functional safety concept requires an analysis of how component faults contribute to hazards. Therefore, ASILs initially assigned to classify the risk posed by hazardous events are further decomposed throughout architectural component faults according to rules described in ISO 26262 Part 9. The benefits of ASIL decomposition are obtained when architectural elements are sufficiently independent. In the case where only two independent components failing together leads to the occurrence of a hazard, the responsibility of addressing an stringent ASIL D assigned to a hazard, is shared between the components (ASIL B + ASIL B). ASIL decomposition allows addressing a higher ASIL assigned to a hazard without being unnecessarily expensive.

ASIL allocation and decomposition are qualitative concepts that address systematic issues (i.e.: design and architecture) rather than random faults (i.e.: hardware reliability). If applied correctly, it allows engineers allocating lower ASILs to components and reusing third party pre-certified parts, while still meeting the safety goals derived from the ASILs assigned to hazardous events [10]. ASIL D is assigned to the most critical hazards/items that demand rigorous assessment process. On the other hand, ASIL QM is usually assigned to hazards/items that pose no safety risks, i.e., not required to satisfy or demonstrate any specific safety goals. ISO 26262 prescribes a set of safety goals, activities, guidance, and work products that should be produced at each phase per ASIL. ASIL D demands more risk reduction measures, e.g., lower failure rates and extensive software verification, compared to ASIL A.

2.2 HiP-HOPS and ASIL Decomposition

HiP-HOPS [19] is a method and tool for model-based safety analysis. HiP-HOPS supports ISO 26262 safety-lifecycle, fault tree analysis and FMEA, via semi-formal languages for specification, composition, and analysis of the system failure behavior based on a set of dependability information about the system components. Once the system models have been annotated with hazards and local failure logic, HiP-HOPS synthesizes fault trees for each hazard, and then combines them to create an FMEA for the system that can record the effect of combinations of component faults.

HiP-HOPS design optimization extension [1] implements ISO 26262 ASIL decomposition rules [10]. HiP-HOPS tool uses the information within fault trees and FMEA results and rationalizes the allocation of ASILs to hazards and their decomposition through system components, by showing how combinations of component failures lead to hazards. The HiP-HOPS design optimization capability was further extended to support ASIL decomposition through components of a variant-intensive system design [16]. This extension supports co-analysis of files containing HiP-HOPS ASIL decomposition results for each system variant, to obtain the ASILs that should be assigned to components to ensure their safe use across a set of variants relevant for the stakeholders. This is achieved by allocating the most stringent ASIL assigned to a failure mode of a component in a given system variant as the required ASIL to ensure the safely use of that component across all variants.

ASIL decomposition results are inputs for deriving ISO 26262 lifecycle process models for individual variant-intensive components without being unnecessarily stringent or expensive. A component process model comprises a set of activities, guidance and artefacts to be produced at each lifecycle phase to comply with the targeted ASIL requirements. Thus, the process of verifying the design of an ASIL C component should comprise design inspection and walkthroughs, control and data flow analyses, and simulation of the dynamic parts of the design to comply with ASIL C safety goals. The verification of the design of an ASIL D component, however, should address other safety goals demanding more costly guidance, e.g., formal verification. Component ISO 26262 life-cycle models can be specified with the support of SPEM 2.0 process modeling tools, e.g., EPF Composer, and their variability can be managed with the support of variant management tools like BVR.

2.3 EPF Composer and BVR

The EPF Composer is a tool built upon the Unified Method Architecture (UMA), which supports the specification and deployment of OMG SPEM 2.0 [17] compliant software process models [6]. UMA incorporates SPEM 2.0 and defines a library for method plugins and configurations. An EPF method plugin is divided into two categories: method content and processes. The method content describes the required steps and skills to achieve specific development goals comprising: content packages, standard custom categories [12]. Therefore, tasks, roles, work products and guidance are specified in a content package, and disciplines, domains, work product kinds, role sets and tools are standard categories. EPF Composer stores all method library content in a repository of XMI files. XMI is an OMG specification for storage and interchanging metadata in XML format. The method content elements are semi-ordered, thus, providing the means to create a process lifecycle. EPF capability patterns are building blocks used for holding process knowledge for a given area of interest, and complete lifecycles are modeled as delivery processes. A method configuration is a subset within a method library. EPF Composer supports the generation of a method configuration as a HTML web page that can be deployed in a web server for distributed collaboration between team members.

BVR [8] is a language and toolset, built upon CVL [9], which supports standard variability modeling in Eclipse Modeling Framework (EMF) models. Since BVR defines variability orthogonally for Meta-Object Facility (MOF) [18] compliant models, e.g., EPF method plugins, communication with other tools is required to map elements of a target configuration to variability abstractions. BVR supports the generation of configurations from a base model via VSpec, Realization and Resolution editors. The *VSpec* editor supports the specification of feature models [3]. A feature is a characteristic of the system visible to the end user. In the *VSpec* model, the mandatory features are connected to the parent feature via solid lines, and dashed lines represent optionality. The VSpec also supports the specification of constraints between features using implication, alternative, and negation operators. The Resolution editor allows engineers resolving variability in a base model to obtain configuration models representing the desired product variants.

The BVR Realization editor supports engineers on mapping variability abstractions (features) to elements of a base model based on placements and replacements within fragment substitution elements. A fragment substitution removes base model elements within placements and substitutes them with replacement elements based on feature selection. BVR [8] provides an intuitive and visual representation where placement and replacement elements are highlighted in red and blue colors respectively. Fragment substitutions are executed based the variability definitions in the abstract (VSpec) and realization layers for deriving configuration models from a base model. The integration of EPF Composer and BVR within the AMASS platform allows mapping EPF method plugin elements to VSpec features in the BVR realization editor, and it supports variability resolution in method plugins.

3 The Proposed Approach

The purpose of our approach is supporting the management of the diversity of safety goals in variant-intensive platform system architectures and safety certification without being unnecessarily stringent or expensive. In this section, we present the structure (Section 3.1) and the steps (Section 3.2) of our approach. We evaluated our approach in an automotive braking system platform.

3.1 Approach Structure

Our approach provides a conceptual framework to manage the diversity of safety goals, supporting the design and safety-certification of variant-intensive platform components based on optimal and cost-effective ASIL allocation results, and automated (re)configuration of standard-compliant component process models. It relies on the integration of model-based safety analysis and ASIL decomposition, software process modeling, and variability management techniques. In this work, we considered HiP-HOPS ASIL decomposition extension for variant-intensive systems and product lines [16], EPF Composer for safety standard process modeling, and BVR for variability management on safety goals and EPF process models. This work enhances AMASS Platform with the support for configuring component process models according to product line ASIL decomposition results provided by HiP-HOPS. In addition to the support for safety certification, our approach intends reduce the burden on configuring process models for components of large-scale and variant-intensive safety-critical software platforms.

Fig. 1 shows an overview of our framework, comprising modules and their relationships in a SysML block diagram. The design of variant-intensive systems can be performed with the support of EAST-ADL [4] or MATLAB/Simulink and pure::variants integration. Preliminary ASIL decomposition for a given product variant is obtained through integration of HiP-HOPS [19] with Simulink or EAST-ADL based tools (e.g., EPM). Cross variant component lifecycle models are generated according to component ASILs provided by product line HiP-HOPS ASIL decomposition extension. The generated process models provide standard-compliant guidance for specification and verification of the design of architectural subsystems and components. Our approach requires the following input artefacts: ASIL decomposition results for a variant-intensive system design (provided by HiP-HOPS), the specification of a superset (150%) process model for the targeted standard, e.g., produced using EPF Composer, the variability specification (VSpec) and realization models for the targeted standard Process Line using BVR tool.

The concept of 150% model relates to the superset approach where different 100% configuration models are obtained, via selection and resolution of variation points from a 150% model containing both base and variable elements [2]. In our case, the 150% model is an EPF/SPEM 2.0 standard Process Line. ASIL decomposition results for a variant-intensive system design are obtained from the analysis of ASIL decomposition results from multiple system configurations (variants) relevant for the stake-

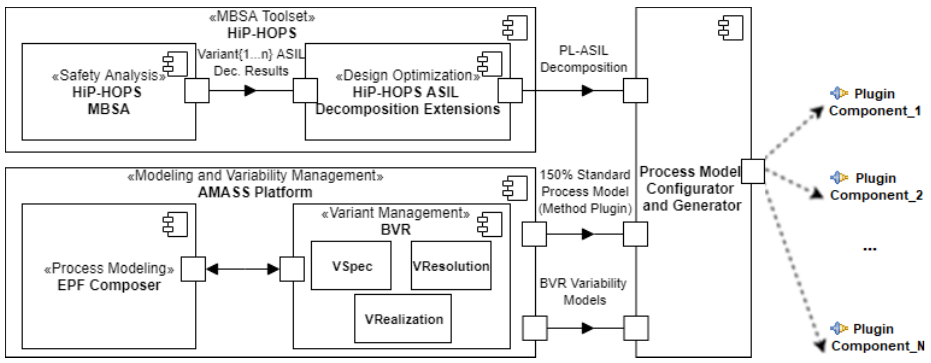


Fig. 1. Tool framework modules and their relationships.

holders. Although the specification of 150% process model and its variability model seems to be a burden, these artefacts can be further reused across different projects and companies.

The ASIL decomposition results, 150% process model for the targeted safety standard, and the BVR variability model are input artefacts to our Process Model Configurator and Generator algorithm[3]. For each system component, our algorithm invokes the BVR API to generate a new resolution model based on the assigned ASIL and it executes the BVR resolution method. We provide the following input parameters to this method: the *VSpec*, *VResolution*, and *VRealization* models for generating an EPF method plugin that only contains the required activities, tasks and guidance to address the ASIL assigned to the component. This is done for each component of the hierarchical platform architecture. If a component contains subcomponents, the same procedure is executed to generate process models for each subcomponent. The algorithm uses a recursive call for configuring and generating process models for architectural subcomponents. This algorithm was implemented in Java and will be further deployed as an Eclipse plugin.

The generated component process models provide development guidance and the basic *claims* for structuring an argument of conformance of component's development processes with safety goals established by the standard for the targeted ASIL. The produced lifecycle artifacts to address the safety goals provide the *evidence* that substantiate *claims* of conformance with safety standards. Component *claims* provided by development lifecycle models and the produced *evidence* can be used for structuring *modular safety arguments* arguing the conformance of component development processes with safety standards, supporting the certification of platform components. Although there are similarities of our approach with what was done in EAST-ADL [4], the issue of variability is addressed more extensively here and it allows ASIL decomposition across a product line which was not done in EAST-ADL.

3.2 Variant Management and EPF Model Configuration/Generation Process

In this section, we describe the steps of our approach considering the ISO 26262 standard and an automotive variant-intensive wheel braking system.

Automotive Hybrid Braking System

The Hybrid Braking System[4] (HBS) [5] comprises one electrical motor per wheel. Fig. 2 shows the HBS architecture in a block diagram. The term hybrid means the braking occurs through the combined action of electrical In-Wheel Motors (IWMs), and frictional Electromechanical Brakes (EMBs). During braking, IWMs transform the vehicle kinetic energy into electricity, which charges the power train battery, increasing the vehicle's range. The HBS architecture comprises 4 variant wheel-brake modules (subsystems), 30 components with 69 connections. Each wheel brake module comprises a Wheel Node Controller (WNC) for calculating the amount of braking

[3] https://github.com/bressan3/HBS-HipHops-Results/tree/master/pseudocode

[4] https://github.com/bressan3/HBS-PL

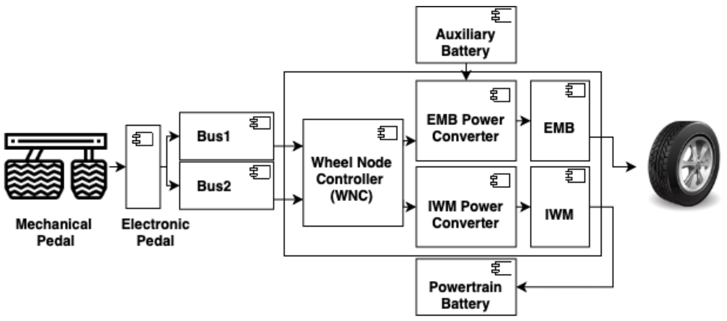


Fig 2. Hybrid braking system architecture [5].

torque to be produced by each wheel braking actuator, and it sends commands to EMB and IWM power converters that control EMB and IWM braking actuators. While braking, the electric power flows from the Auxiliary Battery to the EMB via EMB Power Converter; and IWM acts as a power generator providing energy for the Powertrain Battery via IWM Power Converter.

The wheel brake module is the HBS variation point. We can combine the four HBS wheel-brake modules into different ways to derive different system configurations. The three HBS configurations for generating cost-effective EPF software process models are: four wheel braking (4WB), front wheel braking (FWB), and rear-wheel braking (RWB). The front-wheel brake modules and their connections to other components (Fig. 2) represent the realization of FWB configuration. Different hazards with different ASILs can rise from the interaction between components in each configuration, impacting on ASIL decomposition. FWB and RWB raise two ASIL D hazards each with different causes. 4WB configuration raises two ASIL C and four ASIL D hazards [16].

The Process: Fig. 3 illustrates the approach steps, in an activity diagram, to support variability management on safety goals, configuration and generation of cost-effective EPF process models for variant-intensive software components. The starting points of the process are performing ASIL decomposition for a variant-intensive system design and the specification of 150% EPF process models. These steps can be performed in parallel. After that, we should manage variability on the process model(s) specified with the support of EPF Composer. In our approach, we use BVR for variability specification (i.e., specification of ASIL features) and variability realization (i.e., linking ASIL features to their respective process activities and guidance). Finally, the ASIL decomposition results, along with the EPF process model(s), BVR VSpec and realization models are inputs to the process configurator. Finally, we derive process models for individual variant-intensive components according to their ASILs. We describe the inputs, purpose, and outputs of each step, considering the braking system and ISO 26262 *Part 6-7.4.8.1: System design and verification methods*, as follows.

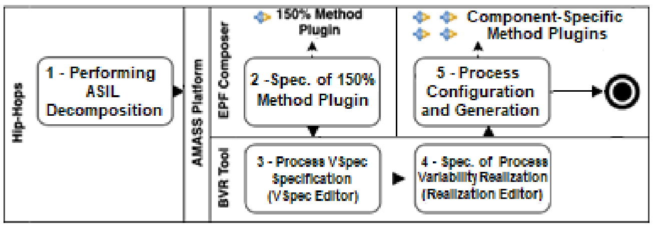

Hip-Hops
1 - Performing ASIL Decomposition
AMASS Platform
EPF Composer
150% Method Plugin
2 -Spec. of 150% Method Plugin
Component-Specific Method Plugins
5 - Process Configuration and Generation
BVR Tool
3 - Process VSpec Specification (VSpec Editor)
4 - Spec. of Process Variability Realization (Realization Editor)

Fig. 3. Steps for configuration and generation of process variants.

1 - Performing ASIL Decomposition in variant-intensive system design: Inputs: a set of ASIL decomposition results for configurations of a variant-intensive system design relevant for the stakeholders, obtained via execution of HiP-HOPS from configuration's fault trees and FMEA. **Purpose**: analyzing ASIL decomposition results from different system configurations to identify the ASILs that should be assigned to ensure the safe use of components across configurations. In our approach, we perform this activity with the support of HiP-HOPS ASIL decomposition extension for variant-intensive system design [16]. We send a set of configuration-specific ASIL decomposition results (XML files) to the HiP-HOPS extensions performing the analysis. **Outputs**: the required ASILs to ensure the safe use of the components across system configurations. Three variants were examined from the HBS and the allocated requirements (i.e., ASILs) to 30 components from which the variants are composed. The possible space allocations that satisfy the safety requirements in each HBS variant design is large, ranging from 1 to 850. The vast majority of those allocations would incur unnecessary costs, i.e., leading to component development at unnecessarily higher ASILs. From the analysis of the results provided by HiP-HOPS, one could see many allocations where costs were higher among those representing good solutions. For example, one allocation solution for a given HBS variant prescribes a stringent ASIL D to the BrakeUnit4.IWM component. Doing the allocation of ASILs manually could incur significantly higher waste of resources. Table 1 shows the best ASIL allocation solutions for HBS components per variant (see columns "4WB", "FWB" and "RWB") provided by HiP-HOPS. We further sent these allocations to the HiP-HOPS extension [16] for analyzing the ASILs assigned to 30 HBS components in each configuration to identify the allocations that ensure the safe use of components across configurations (column "MAX ASIL").

2 - Specification of 150% EPF Process Models: Inputs: the targeted safety standard(s), e.g., automotive ISO 26262, to be modeled using EPF Composer. **Purpose**: specifying the superset (150%) standard process model(s) based on the standard(s)

Table 1. ASIL decomposition results for HBS variant-intensive system components.

Component	MAX ASIL	4WB	FWB	RWB
Auxiliary Battery	D (4)	D(4)	D (4)	D (4)
BrakeUnit1.WheelNodeController	B (2)	A (1)	B (2)	-
BrakeUnit1.EMB Power Converter	B (2)	A (1)	B (2)	-
BrakeUnit4.IWM	B (2)	QM (0)		B (2)

documentation. This steps aims to obtain a superset model(s) with the required processes, phases, activities, tasks, roles, work products and guidance per ASIL. We recommend engineers to follow the Snowball approach [14] rules for analyzing standard(s) and specifying EPF process models. The Snowball rules describe mappings between safety standard concepts and EPF/SPEM 2.0 abstractions. The ISO 26262 *Life-Cycle*, *Section*, and *ASIL recommendation* concepts correspond to SPEM *processes*, *activities*, and *guidance* respectively. In the EPF Composer, a process is a *Delivery Process*, activity is a *Capability Pattern*, and guidance is a *Content Package: Guidance*. Each content package contains optional, recommended, and highly recommended *Tasks*, *Roles*, *Work Products*, and *Guidance* to perform an *Activity* in compliance with all ASILs. **Outputs**: 150% EPF model(s), i.e., ISO 26262 Standard Process Line. We have followed the Snowball approach for specifying the EPF method plugin for the ISO 26262 *Part 6-7.4.8.1: Verification of system design* guidance (illustrated in Fig. 6a). Guidance can be *highly recommended*, *recommended* or *optional* according to the targeted ASIL. We specified a method library with one content package with all the *verification of system design* guidance.

3 - Specification of the Process(es) BVR VSpec Model: Inputs: the specification of superset (150%) standard(s) process models as an EPF method library with the required processes, phases, activities, roles, work products, and guidance per ASIL. **Purpose**: specify the VSpec model, using BVR, for the targeted standard(s) based on the standard guidance and the taxonomy of safety integrity levels. For example, ISO 26262 defines five ASILs: QM, A, B, C, and D and a set of phases, activities, tasks, and recommended guidance to be followed per ASIL. Firstly, we create a *VSpec* model with an ASIL mutually exclusive feature group with the specification of each ASIL as a feature. We should also specify standard processes, phases, activities, tasks and/or guidance as features. For each task, we should create a feature group containing guidance features that represent the optional guidance to comply with each ASIL. Finally, we should specify constraints, using Basic Constraint Language (BCL) from BVR, to establish relationships between a given ASIL and its corresponding guidance feature. **Outputs**: the VSpec model for the EPF standard process model(s) with constraints highlighting the relationships between ASILs and their corresponding processes, phases, activities, tasks, work products and guidance. Fig. 4 shows an excerpt of the VSpec model for the ISO 26262 Part 6-4.7.4.8.1: *Verification of system design*. We created an ASIL feature group with A, B, C, and D mutually-exclusive selection features. We also created features to represent: ISO 26262 Part 6 (process), clauses (tasks) and guidance that may present variability, e.g., Prototype generation and Formal verification features were specified as optional (Fig. 3). We specified constraints to link ASIL features to guidance features, e.g., (C or D) implies (not G+1d).

4 - Specification of Process(es) BVR Variability Realization Model: Inputs: the specification of superset (150%) standard(s) EPF process models and the VSpec model. **Purpose**: specifying mappings linking features in the VSpec model with their realization into EPF method plugin elements, e.g., content packages. For each ASIL/guidance related feature in the VSpec model, e.g., "G_1d", we specify placement and replacement (elements highlighted in blue in Fig. 5b) fragments, and create

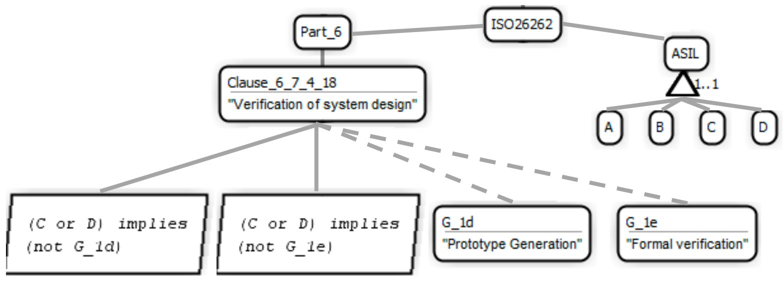
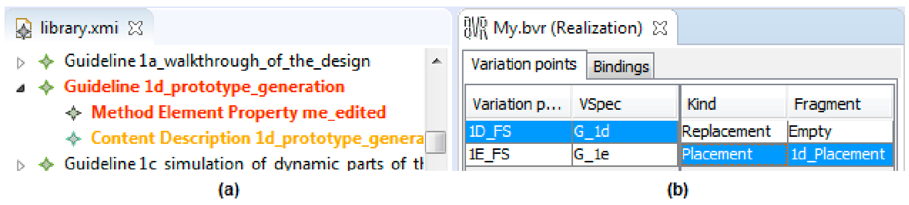


Fig. 4. System design verification VSpec model.

a fragment substitution by following the steps described in Section 2.3. After that, we link the created fragment substitution to a VSpec feature. **Outputs**: the variability realization model with mappings linking VSpec features to their realization into the EPF method plugin elements as illustrated in Fig. 5b. The variability realization model comprises two fragment substitutions that represent the realization of “G_1d” and “G_1e” VSpec features in the EPF process model. When G_1d feature is selected in the resolution model, we generate an EPF method plugin excluding the Prototype Generation guideline (highlighted in red in Fig. 5a) from the method plugin. If ASIL C or D is selected, then “G_1d” and “G_1e” features are chosen.


library.xmi

- Guideline 1a_walkthrough_of_the_design
- Guideline 1d_prototype_generation
  - Method Element Property me_edited
  - Content Description 1d_prototype_genera
- Guideline 1c simulation of dynamic parts of th

My.bvr (Realization)

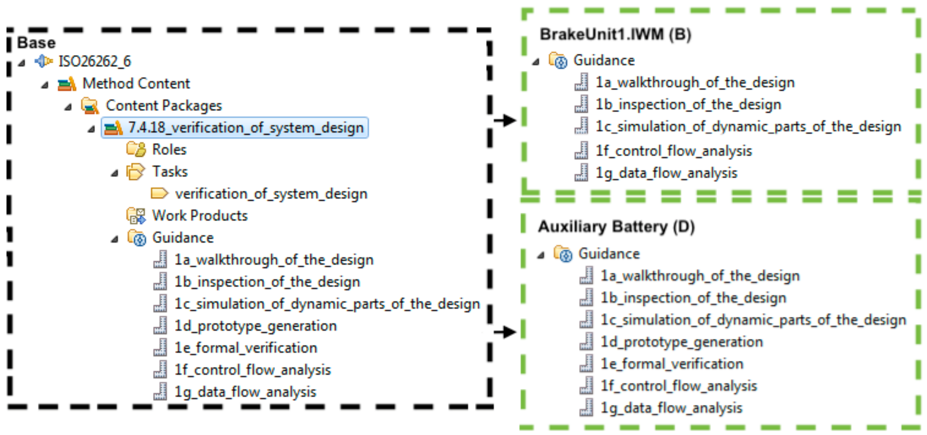
Variation points | Bindings

Variation p...	VSpec	Kind	Fragment
1D_FS	G_1d	Replacement	Empty
1E_FS	G_1e	Placement	1d_Placement

Fig. 5. BVR variability realization model.

5 - Process Configuration and Generation for Variant-Intensive Components: Inputs: ASIL decomposition results for a variant-intensive system design, 150% EPF method plugin, BVR VSpec and realization models. **Purpose**: generating EPF method plugins according to the ASILs assigned to each variant-intensive system component. We do this by providing the four aforementioned inputs to the Process Configurator and Generator program, which analyzes the ASIL decomposition results, and for each component: it selects the proper ASIL feature in the BVR resolution model, and invokes BVR resolution passing the following parameters: VSpec, resolution, realization, and the 150% EPF method plugin. These steps are performed to generate cost-effective EPF method plugins for all system components. **Outputs**: a set of EPF method plugins, one per component. We generated EPF method plugins to 30 HBS components, e.g., ASIL D Auxiliary Battery and ASIL B IWM. Only ASIL D processes contain stringent system design verification guidance. Therefore, we achieved a cost-effective configuration and generation of EPF method plugins for the HBS variant-intensive software components according to their ASILs. Fig 6 shows the base and derived component-specific method plugin models. It is important to highlight that any change in the system design and ASIL decomposition results directly impact on the structure of component's EPF process models. Our approach supports the management of the diversity on safety goals that can emerge from changes in the platform design, via automatic regeneration of EPF process models for components.

The generated EPF process models provide the *claims* for arguing conformance of development processes of individual components with the allocated ASILs. The resultant software development, verification, validation and testing artifacts from process activities, provide the evidence that substantiate *claims* of compliance with safety goals defined for the targeted component ASIL requirements. Process conformance arguments for a given component can be automatically generated, with the support for model-based techniques, from component EPF process model and the respective development artifacts.



```
Base
  ISO26262_6
    Method Content
      Content Packages
        7.4.18_verification_of_system_design
          Roles
          Tasks
            verification_of_system_design
          Work Products
          Guidance
            1a_walkthrough_of_the_design
            1b_inspection_of_the_design
            1c_simulation_of_dynamic_parts_of_the_design
            1d_prototype_generation
            1e_formal_verification
            1f_control_flow_analysis
            1g_data_flow_analysis
```

```
BrakeUnit1.IWM (B)
  Guidance
    1a_walkthrough_of_the_design
    1b_inspection_of_the_design
    1c_simulation_of_dynamic_parts_of_the_design
    1f_control_flow_analysis
    1g_data_flow_analysis
```

```
Auxiliary Battery (D)
  Guidance
    1a_walkthrough_of_the_design
    1b_inspection_of_the_design
    1c_simulation_of_dynamic_parts_of_the_design
    1d_prototype_generation
    1e_formal_verification
    1f_control_flow_analysis
    1g_data_flow_analysis
```

Fig. 6. Base method plugin and the generated component-specific process models.

Considering an excerpt of the EPF process model for BrakeUnit1.IWM component (Fig. 6) to address ASIL B safety goals, we can build a safety argument with a *claim* arguing the "*minimum torque is not violated while braking (ASIL B)*". This *claim* is further supported by *sub-claims* arguing the verification of system design (ISO 26262 Part 4 Sec. 7.4.1.8) was performed following the recommended guidance to address ASIL B. These *sub-claims* argue that the following techniques were applied to verify the BrakeUnit1.IWM design: *walkthrough*, *inspection*, *model simulation*, *control and data flow analyses*. Finally, *design walkthrough* and *inspection*, *control and flow analyses reports* together with *simulation results* provide the *evidence* that substantiate the *sub-claims*. The derived safety argument for each component can be organized into modules.

Component *argument modules* can be further used to support safety certification of the whole configurable platform defined in the HBS design, instead of a specific configuration. A configurable platform allows engineers deriving different variants by combining alternative components. Thus, platform *safety argument modules* provide valid assurance *claims*, supported by *evidence* that demonstrate that the *components* are acceptably safe to operate across a set of targeted *configurations*.

4 Related Work

Existing work on this topic comprises modeling techniques [11, 13, 14] and tools [12], and ASIL allocation and decomposition [16]. Krammer et al. [13] proposed a method content approach based on the EPF Composer to define and formalize software development processes, improving process management and tailoring activities. Their approach supports the tailoring of ISO 26262 lifecycle processes according to ASILs assigned to items, thus, improving reusability and extensibility of method definitions. Our 150% variability modeling approach on EPF process models also considers the impact of ASILs on process activities, tasks, and guidance with the advantage of using only one method content, avoiding the specification of redundant guidance in different method contents as present in the Krammer et al. [13] approach. Krammer et al. also provides mappings between EPF Composer and SPEM elements and ISO 26262 concepts, which are the basis for the "Snowball" [14] approach for extracting SPEM 2.0 process models from standards. Luo et al. [14] propose the "Snowball" approach to support the extraction of the conceptual and process models from safety standards to enable the usage of these models for demonstrating compliance and reusing assurance artefacts. This is a rule-based approach that contributes to reduce the manual work and it provides traceability between conceptual and process models, and the standard. The Snowball approach was applied in the automotive domain for specifying ISO 26262 Part 3 process models using EPF Composer. The approach can also be used to specify process models of standards from other domains, e.g., aerospace DO-178C [22], and industry IEC 61508. The work of Luo et al. is not focused on the generation of cost-effective process models for components as presented in this paper.

Javed and Gallina [12] propose the integration between the EPF Composer and BVR Tool to support variant management and resolution on EPF method libraries, establishing the concept of Safety-oriented Process Lines (SoPL). In [11], Javed et. al. integrated BVR, CHESS Toolset and EPF Composer to support co-engineering and integration of SoPLs and SPLs. The approach supports the specification of traceability links between variability in the software architecture and process elements, and automatic generation of component variants and their respective process models. Although Javed and Gallina consider the association between ASILs and process activities, work products, and guidance in the BVR realization model, the generation of process models for multiple components with different ASILs demands the manual configuration of resolution models, which can be burden in the case of a complex system design. Our approach automates the configuration EPF models based on ASIL assignment. Oliveira et al. [16] propose an extension to the HiP-HOPS design optimization to support ASIL decomposition throughout components of variant-intensive system architectures. Our approach enhances Oliveira et al. [16] work with the support for variability management on safety-oriented process lines and automated configuration of EPF method plugins for components accordingly to their ASILs.

5 Concluding Remarks and Future Work

In this paper, we presented an approach to support variability management on safety goals and semi-automatic configuration and generation of software process models for certification of variant-intensive components. The approach comprises a conceptual framework, tooling integration and a systematic process. In this work, we integrated HiP-HOPS ASIL decomposition extensions [16] for variant-intensive system design, EPF Composer, and BVR tools, to support variability management on safety goals, configuration and automatic generation of cost-effective EPF process models for software components according to ASIL assignment. The process provides a set of steps to support engineers on generating cost-effective EPF process models for individual components based on ASIL decomposition results. This work contributed to reducing the costs and effort for certifying individual components within a system family. The approach automated tailoring of development processes, and enabled component safety certification in compliance with the assigned ASILs, without being unnecessarily expensive. Our approach also enabled the reuse and customization of process models across multiple projects. The generated process models provide the basis for arguing the conformance of component development processes with ASIL requirements. A limitation in our approach is the need for manually creating a 150% EPF process model for the whole standard and a BVR model. As future work, we intend to evaluate our approach in other domains, e.g., aerospace. We also intend to enable support for automatic generation of process-based conformance arguments for individual components from EPF process models. Furthermore, we intend to improve and implement the process configurator algorithm as an Eclipse plugin.

References

1. Azevedo, L. S., Parker, D., Walker, M. Papadopoulos, Y. Automatic Decomposition of Safety Integrity Levels : Optimization by Tabu Search. In: Proc. of the 2nd Workshop on Critical Automotive applications : Robustness & Safety (CARS), Safecomp (2013).
2. Beuche, D., Schulze, M., Duvigneau, M. When 150% is too much: supporting product centric viewpoints in an industrial product line. In: 20th Int. Systems and Software Product Line Conf. (SPLC), pp. 262-269, ACM, Beijing, China, (2016).
3. Capilla, R., Bosch, J., and Kang, K. C. Systems and Software Variability Management: Concepts, Tools and Experiences. Springer Publishing Company (2013).
4. Chen, D. J., Mahmud, N., Walker, M., Feng, L., Lonn, H. Papadopoulos, Y. Systems modeling with EAST-ADL for fault tree analysis through HiP-HOPS. In IFAC Proceeding Volumes (IFAC-PapersOnline), v. 4, pp. 91-96 (2013).
5. De Castro, R., Araújo, R. E., Freitas, D. Hybrid ABS with Electric motor and friction Brakes. In: 22nd International Symposium on Dynamics of Vehicles on Roads and Tracks, Manchester, UK, pp. 1-7, (2011).
6. Eclipse. EPF Composer Manual, http:// https://www.eclipse.org/epf/general/EPF_Installation_Tutorial_User_Manual.pdf, last accessed 2020/02/28.
7. Habli, I. et al.: Challenges of Establishing a Software Product Line for an Aerospace

Engine Monitoring System. In: 11th International Software Product Line Conference (SPLC), pp. 193–202, ACM, Japan (2007).

8. Haugen, Ø., Øgård, O. BVR-Better Variability Results. In: Amyot D., Fonseca i Casas P., Mussbacher G. (eds) System Analysis and Modeling: Models and Reusability. LNCS, vol. 8769. Springer, (2014).
9. Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G. K., and Svendsen, A. Adding Standardized Variability to Domain Specific Languages. In: 12th International Conference on Software Product Lines, Limerick, Ireland, IEEE, pp. 139-148 (2008).
10. ISO. ISO 26262: Road vehicles - Functional safety, (2018).
11. Javed, M. A., Gallina, B., Carlsson, A. Towards variant management and change impact analysis in safety-oriented process-product lines. In Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing, ACM, pp. 2372-2375 (2019).
12. Javed, M. A., Gallina, B.: Safety-oriented process line engineering via seamless integration between EPF composer and BVR tool. In: 22nd Int. Systems and Software Product Line Conf. - Volume B, Gothenburg, Sweden. ACM, pp. 1-6 (2018).
13. Krammer, M., Armengaud, E., Bourrouilh, Q. Method Library Framework for Safety Standard Compliant Process Tailoring, In: 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Oulu, pp. 302-305, IEEE (2011).
14. Luo, Y., Brand, van den, M. G. J., Engelen, L. J. P., Favaro, J., Klabbers, M. D., Sartori, G. Extracting models from ISO 26262 for reusable safety assurance. In: 13th Int. Conf. on Software Reuse, Pisa, Italy, Springer, vol. 7925, pp. 192-207 (2013).
15. Oliveira, A. L., R. T. V., Masiero, P. C., Papadopoulos, Habli, I., Kelly, T Variability Management in Safety-Critical Software Product Line Engineering. In: R. Capilla, C. Cetina, and B. Gallina (Eds.), ICSR 2018, LNCS 10826, pp. 1–20 (2018).
16. Oliveira, A. L., Braga, R. T. V., Masiero, P. C., Papadopoulos, Y., Azevedo, L., Parker, D., Habli, I., Kelly, T. Automatic allocation of safety requirements to components of a software product line. In: 9th IFAC Symp. on Fault Detection, Supervision and Safety for Technical Processes, Paris, France, Elsevier, v. 48, i. 41, pp. 1309-1314 (2015).
17. OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) Ver 2.0, http://www.omg.org/spec/SPEM/2.0/, last accessed 2020/02/25.
18. OMG. Meta-Object Facility, http://https://www.omg.org/mof/,last accessed 2020/03/01.
19. Papadopoulos Y., Walker M., Parker D., Rüde, E., Hamann, R., Uhlig, A., Grätz, U., Lien, R. Engineering failure analysis and design optimization with HiP-HOPS. Journal of Engineering Failure Analysis, Elsevier. 18 (2), 590-608, (2011).
20. Parker, D., Walker, M., Azevedo, L., Papadopoulos, Y., Araujo, R. Automatic Decomposition and Allocation of Safety Integrity Levels Using a Penalty-Based Genetic Algorithm. In: Recent Trends in Applied Artificial Intelligence, LNCS, vol. 7906, pp. 449-459, Springer-Berlin, Heidelberg (2013).
21. Pohl, K., Böckle**,** G., van der Linden, F. J. Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005).
22. RTCA: DO-178C Software Considerations in Airborne Systems and Equipment Certification. Radio Technical Commission for Aeronautics (2011).
23. S.A.E. ARP 4754A: Guidelines for development of Civil Aircraft and Systems. (2010).
24. Schulze, M., Mauersberger, J., Beuche, D. Functional safety and variability: can it be brought together? In: 17th Int. SPLC, 236-243, ACM, NY, USA (2013).