VROOM & cC: a Method to Build Safety Cases for ISO 26262-compliant Product Lines

Barbara Gallina[1], Antonio Gallucci[1], Kristina Lundqvist[1], and Mattias Nyberg[2]

[1] Mälardalen University, P.O. Box 883, SE-72123 Västerås, Sweden
name.surname@mdh.se
[2] Scania AB, Södertälje, Sweden
mattias.nyberg@scania.com

Abstract. ISO 26262 is a functional safety standard that targets the automotive domain. This standard focuses on single system certification[1] and does not contain guidelines to certify product lines. Thus, to be ISO 26262-compliant, for each product of a product line, a company must provide almost from scratch all the work products required by the standard, including a safety case. Current product lines engineering methods represent an effective solution to systematize reuse. These methods, however, are not aligned with safety standards and thus they lose their strength when adopted to engineer safety-critical product lines. To enable and accelerate systematic reuse, we introduce VROOM & cC, a new method that by integrating traceable management of commonalities and variabilities at each step of the first two phases of the ISO 26262 safety life-cycle as well as at each stage of the safety case life-cycle permits safety managers to argue about functional safety of product line members by reusing argumentation fragments. To illustrate our approach, we consider a small-sized safety-critical product line.

Keywords: ISO 26262, Safety-critical product lines, Reusability, Variability management, Families of safety cases, GSN for product lines

1 Introduction

Product lines (PLs) represent sets of products that exhibit significant common characteristics and thus can be developed concurrently in order to systematize and maximize reuse. Within the automotive domain, PLs can be identified very frequently. The road vehicles (e.g. cars) for instance may be treated as a PL. Similarly, the majority of the sub-systems that compose road vehicles may be treated as PLs. These PLs, however, exhibit one additional common characteristic: they all have to guarantee a certain level of functional safety. Some classes

[1] In this paper, we use the term *certification* to denote demonstration of functional safety of a given product in compliance with a given standard.

of road vehicles are already expected to comply with ISO 26262 [1] (the recently issued functional safety standard), some others (e.g. trucks and buses) are expected to be compliant with it only by 2016 [2].

The traditional Product Line Engineering (PLE) methods are not aligned with safety standards and thus they are not adequate to engineer safety-critical PLs. Similarly, ISO 26262 mainly addresses single product development and certification and no guidelines are at disposal to address PLs. As a consequence, currently, very little reuse can be systematically planned since no methodological framework is available. To overcome this lack, we propose to align the ISO 26262 safety life-cycle as well as the typical safety case life-cycle with traditional PLE methods. More specifically, we propose a new method called VROOM (Vehicle Road Obligation Oriented Method) & controlled CRASHES (Claim Reuse for Arguing Safely on Hazards Elimination and or Softening). VROOM & cC allows users to specify, trace, and manage commonalities and variabilities within work-products at each step of the first two phases of the ISO 26262 life-cycle as well as safety case life-cycle. Since these work-products are used as evidence for certification purposes to claim for functional safety, by specifying what is common and what varies within them, we also specify what is common and what varies within certification artefacts (i.e. safety case). Thus, from a PL-oriented certification artefact it is easy to derive/build certification artefacts for single products by reusing what is common in combination with adequate variations. To show the effectiveness of VROOM & cC as a reuse catalyst, we apply it to a small-sized safety critical PL (a simplified version of the one considered in [3]).

The rest of the paper is organized as follows. In Section 2 we recall essential background. In Section 3, we present VROOM & cC and in Section 4 we apply it to the small-sized safety critical PL. In Section 5, we present some related work. Finally, concluding remarks and future work can be found in Section 6.

2 Background

In this section we recall the essential background on which we base our work. In Section 2.1, we recall the basic principles of PLE. In Section 2.2, we give an overview of ISO 26262. In Section 2.3, we recall the definition of safety case and how sets of safety cases can be documented. Finally, in Section 2.4 we introduce the safety-critical PL to be developed in compliance with ISO 26262.

2.1 Product Lines Engineering (PLE)

A (Software) PL is a set of (software-intensive) systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [4]. PLE methods deal with the concurrent engineering of a family of products. These methods are characterized by two phases: domain engineering and application engineering. In the first phase, commonalities and

variabilities are engineered; while in the second one, they are reused to derive single products.

To make the paper self-contained, in what follows we provide some basic definitions used within the PL community. A *commonality* is the ability of an asset to be maintained as a constant; while a *variability* is the ability of an asset to be changed (customized) for use in a particular context. Variabilities are defined through variation points and variants. A *variation point* is the place within an artefact where a decision can be made [5]. *Variants* represent the alternatives associated to a variation point. A *Feature Diagram* (FD) is a graphical representation of a hierarchically arranged set of *features* (properties of a system that are relevant to some stakeholder and are used to capture commonalities or discriminate among systems in a family). Features might be mandatory (in case of commonalities), optional or alternative (in case of variabilities).

2.2 ISO 26262-compliant Safety-critical Systems Engineering

The ISO 26262 functional safety standard [1] provides appropriate development processes, requirements and safety integrity levels, called ASILs, specific for the automotive domain. More specifically, ISO 26262 addresses the functional safety of electrical/electronic and programmable electronic safety-related systems within road vehicles with a maximum gross weight of 3.5 tons.

To develop a system in compliance with the standard, the development process shall be adopted and all the work-products shall be provided, including a safety case. The reference system development process model mandated by the standard is a V-model, which consists of three phases: Concept, Product development, and Production and operation phases. In this paper, we mainly focus on the first two phases, which are briefly recalled in what follows.

Concept phase (part 3)- During this phase, the item which could be a system or array of systems is firstly defined. This entails in defining the requirements of the item as well as the purpose, the primarily assumed architecture, boundary and interfaces of the item. Secondly, potential hazards of the item are identified and ASIL-classified through hazard analysis and risk assessment, by considering the severity, controllability and exposure combinations. Concurrently, *Safety Goals* (SGs), which inherit the ASILs of the corresponding hazards, shall be formulated for the identified hazards. These SGs describe requirements needed to avoid hazards or to reduce the risk associated with hazards to an acceptable level. Thirdly, *Functional Safety Requirements* (FSRs) shall be specified for each SG. FSRs describe basic safety mechanisms, implementation- independent safety-related behaviour, and safety measures that have to be provided by elements in the primarily assumed system architecture for complying with the SGs and their ASILs. FSRs only consider functional aspects of the system and not how these are technically implemented in hardware or software. FSRs inherit the same ASILs as the corresponding SGs and shall be allocated to elements of the primarily assumed system architecture.

Product development phase (part 4, 5 and 6)- This phase includes activities that pertain to the entire system, the hardware and the software. In this pa-

per, we limit our attention to the activities related to the entire system (part 4). At the system level, *Technical Safety Requirements* (TSRs) shall firstly be specified for each FSR. TSRs shall detail and decompose the FSRs into requirements which describe how to implement the safety mechanisms described by the FSRs. Secondly, a system design that meets the FSRs and TSRs shall be developed. Hence, the design must be verified against the TSRs. Verification may be conducted using different methods e.g. Fault Tree Analysis, FTA.

ISO 26262 mainly targets single systems. However, to deal with series of systems, the *Safety Element out of Context* (SEooC) concept has been introduced. SEooC denotes reusable subsystems or components which are not yet items and which are developed without having in mind particular vehicle or customer. Despite the introduction of this concept, reuse is not systematically supported.

2.3 Families of Safety Cases

To make the paper self-contained, we partly recall what was summarized in [6] concerning safety cases for single products. The main focus of this subsection, however, is on families of safety cases (that we call safety case lines) aimed at arguing about functional safety of PLs.

A *safety case* is a contextualized structured argument containing process and product-based arguments to link evidence with claims. Process-based arguments show that the system has been developed in compliance with the processes mandated by the standards. Product-based arguments show that the product satisfies the safety requirements derived during the hazards analysis. The purpose of a safety case is to show that a system is acceptably safe. A safety case should not be provided at the end of the development life-cycle. Its provision should be aligned with the system development life-cycle and should go through at least three (preliminary, interim, and operational) stages [7]. In the context of PLs, a safety case exhibits commonalities as well as variabilities, thus it represents a family of safety cases. Variabilities are classified into *intrinsic* (a variation due to style variation used for arguing) and *extrinsic* (a variation due to a product variation) [8]. In this paper we focus on the extrinsic ones.

To document safety cases, Goal Structuring Notation (GSN) [9] represents a promising means. GSN, in particular, permits users to structure their argumentation into flat or hierarchically nested graphs (constituted of a set of nodes and a set of edges), called goal structures. Since GSN has limited modelling capabilities, recently it has been extended to support variability modelling to document safety cases of PLs [8]. This extension is of particular interest in the context of this paper since it allows safety managers to document extrinsic variability, which makes evident the impact onto the goal-structure due to the choices performed during the application engineering phase. Thus, by introducing support for variability, GSN offers implicitly the possibility to model a set of safety cases (a safety case line). In Figure 1, we recall the concrete syntax of some core and extended GSN modelling elements used in Section 4. In what follows, we also briefly recall their informal semantics. The interested reader may refer to [8, 9] for a complete introduction of GSN and its extensions: *Goal*: represents a claim

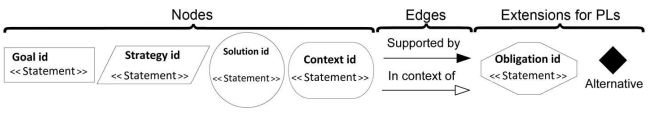

Fig. 1. Subset of extended GSN modelling elements for PLs.

about the system. *Strategy*: represents a method that is used to decompose a goal into sub goals. *Solution*: represents the evidence that a particular goal has been achieved. *Context*: represents the domain or scope in which a goal, evidence or strategy is given. *Supported by*: represents an inferential or evidential relationship. Inferential relationships declare that there is an inference between goals in the argument. Evidential relationships declare the link between a goal and the evidence used to substantiate it. *In context of*: represents a contextual relationship. *Obligation*: represents a condition related to a variation point due to intrinsic or extrinsic variability. In section 4 we use the obligation for extrinsic variability. *Alternative*: represents a XOR choice between different argumentation fragments. As Figure 1 shows, all the nodes as well as the obligation element are characterized by an identifier (ID) and a statement in natural language.

2.4 PL of Fuel Level Estimation and Display Systems

The PL under examination is called Fuel Level Estimation and Display Product Line (FLEDPL). Each single product (Fuel Level Estimation and Display System, shortened FLEDS [6]) is part of a system of systems aimed at providing the controlling functionalities needed in heavy trucks and buses produced by Scania. Thus each FLEDS plays a significant role in terms of global system safety.

The main common functionality characterizing all products of FLEDPL consists of the estimation of the fuel level in the vehicle tank and presentation of this level on the display located in the dashboard. Additionally, the driver must be warned if the fuel level is below a predefined level. As Figure 2 shows, these functionalities are deployed onto different ECUs (Electronic Control Units): Instrument Cluster (ICL), Coordinator (COO), and Engine Management System (EMS). In Figure 2, a filled bullet describes that an ECU is mandatory, while an empty bullet describes that an ECU is optional. Thus, ICL and COO are mandatory elements of the PL while EMS is an optional one, depending on the vehicle type (truck or bus). EMS is responsible for calculating the fuel consumption rate. COO is responsible for estimating the fuel level and determining if the fuel level is low. To reduce noise (due to fuel fluctuations within the tank), COO provides the estimation by filtering the fuel level using a Kalman filter in case of trucks and a low-pass filter in case of buses. To do that, in case of trucks, it uses the information provided by EMS. Finally, the ICL is responsible for displaying the fuel level and activating the low fuel level warning, if needed. The ECUs are interconnected by a CAN (Controller Area Network) bus.

Each product of the PL has one sensor located in the fuel tank to sense the fuel volume and one actuator (fuel gauge) for displaying the fuel level to

the driver. Moreover, a lamp is used to warn the driver in case a low fuel level has been reached. Products may differ with respect to various characteristics e.g. tank capacity, tanks number, fuel type, vehicle type onto which they are deployed. In this paper, we focus on the last characteristic and its implication in the way the common functionalities are consequently differently developed.

The correct behavior of each FLEDS or at least a behavior qualifiable as acceptably safe is necessary to contribute to the functional safety of the global system. For instance, in the case of low fuel level, if the driver is not alerted (omission failure), it may happen that suddenly the vehicle (truck or bus) stops and, in case of normal traffic conditions, human life can be threatened.

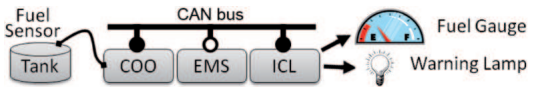

Fig. 2. Fuel Level Estimation and Display Product Line.

3 VROOM & controlled CRASHES

As recalled in the background, PLE represents a valid solution to enable and speed up reuse. However, when the PL is constituted of safety-critical systems, PLE must be adapted [10] to take care of safety-related activities required for certification purposes. As mentioned in the introduction and as experienced while acquiring familiarity in building safety cases [6], ISO 26262 does not provide systematic support for PL development and certification. As a consequence, to be ISO 26262-compliant, a company that manufactures a set of products that can be treated as a PL tends to re-invent the wheel each time a safety case for single products is needed. To enable and accelerate the reuse of development and argumentation artefacts, we propose a new method called VROOM & cC. The main idea of VROOM & cC is to align PLE method with the ISO 26262 safety life-cycle as well as with the safety case life-cycle. To do that we propose a double V reference model for both life-cycles (development and safety case). The double V is motivated by the integration of the two-phase PLE method. Figure 3 shows the double V for the development life-cycle. At each level of the first two phases of the ISO 26262 safety life-cycle, we propose to focus on the domain engineering phase first and then on the application engineering phase. Based on literature results, at each level, we propose to adopt appropriate PLs-oriented modelling capabilities, allowing for specification and management of what varies and what remains constant. Whenever, no modeling capabilities supporting PLs are at disposal, we extend the well-known single product-oriented ones to overcome the lack. The key idea is to include placeholders for the points that may vary and at the same time indicating the corresponding obligations (which, as recalled in Section 2.3, represent conditions related to either intrinsic

or extrinsic variability). For this reason we use the *obligation* term to name our approach. Thanks to these placeholders and their corresponding obligations, it is possible to provide a management means over all variation points in the development and certification life-cycle allowing for navigation and resolution. For space reasons, Figure 3 only sketches the proposed PLs-oriented modelling capabilities. These modeling capabilities are partially used in Section 4 and thus their PL-oriented features become clear during their usage.

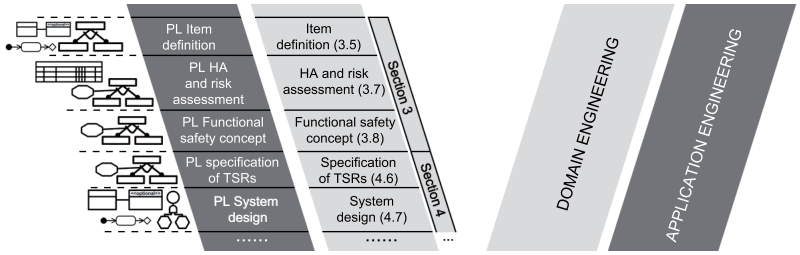

Fig. 3. ISO 26262 safety life cycle and PLE alignment in VROOM & cC.

As Figure 3 shows, to define the item(s), we propose the adoption of FDs, as well as extended SysML block and activity diagrams [3]. More specifically, FDs are used to model the usage context and the PL requirements; while SysML block and activity diagrams are used to define respectively the structure and behaviour of the PL (preliminary architecture). To analyze the hazards and assess the risk, we use an extended HAZard and OPerability analysis (HAZOP). As proposed in the context of the course Safety-critical systems engineering [11], we consider additional fields (namely, exposure, controllability, severity, and ASIL) in order to be compliant with ISO 26262. Moreover, to deal with PLs, we add two fields: one called *Obligation* and the other one called *Variation points*. We use the first one to specify, trace and provide management support for interdependencies of variants selection. We use the second one to specify variation points within the HAZOP analysis. Variation points are specified by indicating the type of variation, which can be either optional (OPT) or alternative (ALT), as well as the data that may vary with the possible values. To define the SGs as well as FSRs and TSRs, we propose to use FDs enriched by obligations. To design the architecture, we propose to extend SysML block and activity diagrams in a similar way as done with other modeling capabilities. To conduct safety analysis at the architectural level, since within the safety community, safety analysis techniques have been extended to reason about PLs, we propose to use [12] but we rename the *condition* modeling element into *obligation* for naming conventions and consistency within our approach. To document the safety case line, we propose to use the extended GSN (which was partially recalled n Section 2.3) at each stage of the safety case life-cycle.

All the models that constitute the work-products of the safety development life-cycle and of the safety case life-cycle are not disconnected. Inclusion/exclusion dependencies between variation points placed within different models are carefully traced, either by using the Obligation symbol or by using semantic equivalents. Traceable management of variabilities allows VROOM & cC users to accelerate controlled Claim Reuse for Arguing Safely on Hazards Elimination and or Softening (CRASHES).

4 Applying VROOM & cC to the PL of FLEDS

In this section, we partially apply VROOM & cC to the small-sized product line which was briefly introduced in Section 2.4. The purpose of this section is to show the effectiveness of VROOM & cC as a reuse catalyst. For space reasons, we only present some work-products obtained by applying VROOM & cC. The interested reader may find a complete illustration in [3].

As indicated in Figure 3, to apply VROOM & cC, first of all we must define the item. The FD, shown in Figure 4, models the usage context. The FDs related to the PL requirements as well as the SysML block and activity diagrams related to the preliminary architecture can be found in [3]. Figure 2, however, in Section 2 provides an informal description of the preliminary architecture.

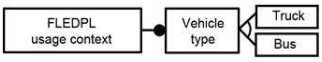

Fig. 4. Usage Context.

In the context of this paper, the usage context models the deployment context. Each FLEDS is deployed either onto a bus or onto a truck. In general, a choice related to the variation point in the usage context has a significant impact: it limits the choices that can be performed at the variation points related to requirements, hazard analysis, system architecture and so on.

The second step to be performed is PLs-oriented hazards analysis and risk assessment. The result of the adapted HAZOP is shown in Figure 5. As Figure 5

ID	Obligation	Item	Attribute	Guide word	Operational situation	Deviation	Consequences	S	E	C	ASIL	Possible Causes	Variation points
H1	**if Vehicle type=Truck then V1.Filter=Kalman else V1.Filter=Low-Pass**	Fuel level estim.	Value	Higher	Highway road, good visibility, normal traffic, bad road conditions (i.e. snow), >40Km/h	Indicated fuel level > actual fuel level	Unpredictable lack of fuel leading to severe injuries (due to e.g. front/reare collision with other vehicles)	S3	E3	C3	C	**{V1}** filter failure	**V1: Type = {Alt}, Filter = {Kalman, Low-Pass}**

Fig. 5. Adapted HAZOP.

reports, we identify one hazard, named "unpredictable lack of fuel" and due

to its values in terms of severity, controllability and exposure, according to the guidelines of ISO 26262, we classify it as ASIL C. This hazard is caused by the deviation of the fuel level estimation function. As the variation point {V1} and the Obligation field indicate, this deviation may be caused either by a failure in the Kalman filter (in the case of trucks) or by a failure in the Low-Pass filter (in the case of buses). Thus the impact of the choices performed in the usage context is clearly traced within our adapted HAZOP.

Figure 6 shows SGs, FSRs and TSRs for the PL aimed at satisfying parts (3-4) of the PL-aligned ISO 26262 development life-cycle. As Figure 6 shows, concerning SGs and FSRs there is no variability. Instead, depending on the vehicle type the FSR called FSR_1.1 is refined in a different way. In case of trucks, FSR_1.1 is refined by TSR_1.1.1, while in case of buses by TSR_1.1.2. This is due to the type of the filtering algorithm that is used. The definition of the PL architecture using SysML block and activity diagrams as well as the verification of TSRs by conducting hazard analysis, more specifically FTA technique can be found in [3]. Verification has also been conducted through simulation.

According to ISO 26262, a safety case should compile the work products that are generated during the safety development life cycle. In this section, our aim is not to present a complete safety case line but to show how the reusable development life-cycle-related work-products can be compiled to achieve argumentation fragments to be integrated in the safety case line. Our safety case line can be considered as a work-product of the safety case line life cycle at the interim stage. A fragment of the safety case line is presented in Figure 7. The left part of the fragment starts by instantiating the hazard avoidance pattern [13]. It, thus, claims (G1) that each FLEDS is acceptably safe, given a definition of acceptably safe. G1 is then refined into G2, by using the strategy S1 (i.e. by arguing by claiming that all the identified hazards have been addressed). G2 is then broken down into G3 and G4. G4 is directly supported by the solution E1. G3 is broken down either into G5 or into G6, depending on the filter type, which in turn depends on the usage context variation point, as expressed in the obligation O1. Finally, G5 and G6 are supported by direct evidence (E2 and E3). From Figure 7 it clearly emerges what is common (directly reusable) and what varies (partially reusable). Variants are clearly located as well as their dependencies with extrinsic variabilities. From this safety case line, thus, safety cases for single products can be easily derived. In particular, two safety cases can be derived which only differ in the way G3 is refined (see alternative).

5 Related Work

Some studies have been conducted on adapting PLE method for safety-critical systems and/or enabling reuse in safety certification. In [10], authors investigate how to align PLE with the avionic standard. They propose a meta-model to capture the entities involved in PL certification and the relationships among them, such as usage context, features, development activities to be performed and the level of certification to be achieved. Authors mainly limit their attention to the

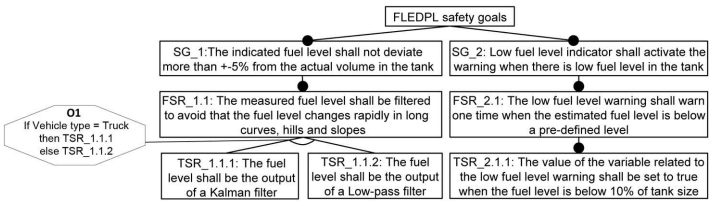

Fig. 6. SGs, FSRs, TSRs.

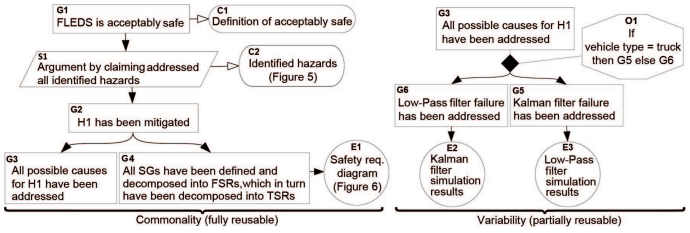

Fig. 7. Fragment of the Safety Case Line.

process aspects, i.e. the impact of a specific safety level onto the process activities to be performed. In our work, we go beyond it since we also consider the impact onto safety cases and we specify and manage how a different choice within an FD influences the goal structure that documents a safety case. As mentioned in Section 2, a GSN extension has been proposed to document safety cases addressing product lines [8]. The authors show the usage of their extension by binding the variation point related to the design choices with the corresponding variation point related to the variants in terms of argumentation fragments within the goal structure. In our approach, we use their GSN extensions and we explore various binding possibilities (beyond design choices). In [14] a model based approach combined with a product line approach is considered. The intention of including the definition of the safety case in the product line process is expressed, but no approach is presented. In [15], authors examine how a safety case can be developed for the assurance of the ISO 26262 functional safety concept. Authors express the need of tracing the various models (e.g. context and feature models, SysML models, and GSN goal structures) and give an intuition about how this could be carried out for single systems. They also mention system variations but they do not treat system variations according to a PL approach. In our work, we also focus on the ISO 26262 functional safety concept, but in addition we consider the entire Concept phase and part of the Product development phase. Moreover, we explicitly consider PLE methods and their alignment with ISO 26262 and we propose how to model, trace and manage variability for each step that we considered. In [16], authors focus their attention on safety-critical PLs in the avionic domain and analyze the relations among usage context and certification levels. Their objective is to trace how certification issues influence PL development in terms of activities to be performed. Our work also focuses on

PLE for safety-critical systems. However, we focus on the automotive domain and we take the alignment a step further since we also consider the impact on safety cases. In [17] a language, called EAST-ADL is presented. This language offers constructs for specifying automotive features at different abstraction levels (e.g. vehicle, analysis, design, etc.), which recently in the context of ATTEST2[2] have been aligned with ISO 26262. A specific extension is available to manage variability in case of PLs. Variants can be selected at variation points and configuration links (which play the role of our *obligation* element) allow these choices to be traced throughout the development life-cycle. PL modeling/safety argumentation and variability management, however, are not supported at each step of the development and safety case life cycle (i.e. hazards analysis and safety cases remain single product-oriented). VROOM & cC, instead, aims at guiding (safety) engineers in systematizing reuse at each step.

6 Conclusion and Future Work

Recently, in the automotive domain, the ISO 26262 standard has been introduced to ensure the functional safety of safety-critical systems. This standard targets the development and certification of single products but does not provide guidelines to treat PLs. As a consequence reuse of artefacts is hindered. To enable and accelerate reuse of development as well as certification artefacts, in this paper we have proposed a novel approach called VROOM & cC. Our approach aligns best development methods in (safety-critical) PLs engineering with ISO 26262. More specifically, we have proposed how to specify, manage and trace commonalities and variabilities at each step of part of the ISO 26262 safety life-cycle and part of the safety case life-cycle. More precisely we have limited our attention to the concept phase and part of the product development phase. To show the effectiveness of our approach as reuse catalyst, we have considered a small-sized safety-critical PL.

As mentioned in the background, the examined small-sized PL is part of a bigger PL (constituted of the entire vehicles). In the mid-term future, to evaluate the scalability of our approach, we aim at applying it to a more complex PL (e.g. the entire instrument cluster). The ultimate goal would be to apply VROOM & cC to the PL constituted of Scania vehicles. Moreover, inspired by [10] we intended to analyze in which way features contribute to safety (work against, work for, etc) to be able to better ponder ate their selection. In the mid-term future, we also plan to cover the entire ISO 26262 safety life-cycle and thus propose techniques to specify, manage and trace commonalities and variabilities within the remaining phases. Currently, VROOM & cC consists of a set of guidelines, in a long-term future, we plan to provide a tool-chain to allow users to apply VROOM & cC (semi) automatically.

Acknowledgments. This work has been partially supported by the Swedish SSF SYNOPSIS project [18].

[2] http://www.atesst.org

References

1. ISO 26262: Road vehicles – Functional safety. International Standard (Nov. 2011)
2. LinkedIn Group: ISO 26262 Functional Safety
3. Gallucci, A.: Building a safety case for a small-sized product line of Fuel Level Display Systems. Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden (to appear in 2013)
4. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison Wesley, Reading, MA, USA (2001)
5. Gurp, J.V., Bosch, J., Svahnberg, M.: On the notion of variability in software product lines. In: Proc. of the Working IEEE/IFIP Conference on Software Architecture (WICSA), Washington, DC, USA, IEEE Computer Society (2001) 45–54
6. Dardar, R., Gallina, B., Johnsen, A., Lundqvist, K., Nyberg, M.: Industrial Experiences of Building a Safety Case in Compliance with ISO 26262. In: Proc. of the 2nd WoSoCER, joint event of the 23rd International Symposium on Software Reliability (ISSRE), Dallas, Texas, USA (Nov. 2012) 349–354
7. Kelly, T.: A Systematic Approach to Safety Case Management. In: Proc. of the SAE World Congress, Detroit, USA, Society for Automotive Engineers (Mar. 2004)
8. Habli, I., Kelly, T.: A Safety Case Approach to Assuring Configurable Architectures of Safety-Critical Product Lines. In: Proc. of the International Symposium on Architecting Critical Systems (ISARCS), Prague, Czech Republic, Springer (Jun. 2010) 142–160
9. GSN Community: GSN Community Standard Version 1 (Nov. 2011)
10. Braga, R.T.V., Trindade Junior, O., Castelo Branco, K.R., Neris, L.D.O., Lee, J.: Adapting a Software Product Line Engineering Process for Certifying Safety Critical Embedded Systems. In: Proc. of the 31st Conference on Computer Safety, Reliability, and Security (SAFECOMP), Springer (Sep. 2012) 352–363
11. DVA321: Safety Critical Systems Engineering. Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden, http://www.mdh.se/studieinformation/VisaKursplan?kurskod=DVA321
12. Dehlinger, J. and Lutz, R. R.: Software Fault Tree Analysis for Product Lines. In: Proc. of the 8th IEEE International Symposium on High Assurance Systems Engineering (HASE), Tampa, Florida (Mar. 2004)
13. Kelly, T., Mcdermid, J.A.: Safety Case Construction and Reuse using Patterns. In: Proc. of the 16th International Conference on Computer Safety and Reliability (SAFECOMP), York, UK, Springer-Verlag (1997) 55–69
14. Burton, S. and Habermann, A.: Automotive Systems Engineering und Functional Safety: The Way Forward. In: Proc. of Embedded Real Time Software and Systems, Tolouse, France (Feb. 2012)
15. Habli, I., Ibarra, I., Rivett, R., Kelly, T.: Model-Based Assurance for Justifying Automotive Functional Safety. In: Proc. of the SAE World Congress, Detroit, Michigan, USA (Apr. 2010)
16. Braga, R.T.V., Trindade Jr, O., Branco, K.R.L.J.C., Lee, J.: Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In: Proc. of the 16th International Software Product Line Conference (SPLC) - Volume 1, New York, NY, USA, ACM (2012) 249–258
17. EAST-ADL Specification: http://www.atesst.org/home/liblocal/docs/ATESST2_D4.1.1_EAST-ADL2-Specification_2010-06-02.pdf
18. SYNOPSIS-SSF-RIT10-0070: Safety Analysis for Predictable Software Intensive Systems. Swedish Foundation for Strategic Research