A Model-Based Approach to Support the Automatic Safety Analysis of Multiple Product Line Products

André L. de Oliveira[1], Rosana T. V. Braga[1], Paulo C. Masiero[1], Yiannis Papadopoulos[2], Ibrahim Habli[3], Tim Kelly[3]
[1]Mathematics and Computer Science Institute, University of São Paulo, São Carlos-SP, Brazil
[2]Department of Computer Science, University of Hull, Hull, United Kingdom
[3]Department of Computer Science, University of York, Deramore Lane, York, United Kingdom
{andre_luiz, rtvb, masiero}@icmc.usp.br, y.i.papadopoulos@hull.ac.uk, {ibrahim.habli, tim.kelly}@york.ac.uk

Abstract—Software product lines (SPL) have been successfully used in the development of automotive and avionics critical embedded systems. Hazards and their causes may change according to the selection of variants in a particular SPL product. Thereby, lower-level assets like fault trees and FMEA (Failure Modes and Effects Analysis) cannot be reused because they are dependent upon the selection of product variants. In this paper, model-based safety analysis techniques and SPL variability management tools are used together to reduce the effort of product safety analysis by: reusing SPL hazard analysis, and providing automatic safety analysis for each SPL product. Therefore, we propose a model-based approach to support the generation of safety analysis assets for multiple safety-critical SPL products. The proposed approach is illustrated using the Hephaestus variability management tool and the HiP-HOPS model-based safety analysis tool to generate fault trees and FMEA for the products of an automotive hybrid braking system SPL. Applying the approach reduced the effort to perform product safety analysis.

Keywords—safety-critical product lines; product, model-based safety analysis.

I. INTRODUCTION

Critical embedded systems are computer systems ranging from small devices to complex monitoring and process management systems. Response time and worst case execution time are important design concerns in these systems, and they should also satisfy safety, reliability, and availability requirements [1]. Failures in critical embedded systems can lead to catastrophic consequences causing injuries or the loss of human's life.

Software Product Lines (SPL) have been successfully used in the development of critical embedded systems in automotive [2] and avionics [3] domains. SPL [4] is an integrated approach in which early life-cycle development and assessment artefacts such as requirements and analysis models can be reused as long as they adhere to the context, architectural constraints and variation rules defined in the SPL. The ability to reuse safety analysis, and not just implementation assets, is important for safety-critical product lines [5] by reducing the effort in performing product safety analysis. Otherwise, the value of a safety-critical SPL can be easily undermined if there is a need to derive fault trees and FMEA analysis by performing safety analysis from scratch, or in an ad-hoc manner, for each SPL product.

Variation in safety-critical product lines is spread throughout architecture, hazards and their causes. As safety is context-dependent, hazards and their causes are not simple to reuse, as they may change according to the selection of product variants. Moreover, it is not possible to reuse low-level analysis assets such as Fault Tree Analysis (FTA) and FMEA results, because they are also dependent upon the selection of product variations. Existing techniques to support product line safety analysis such as Software FTA [6][7][8][9] are not sensible to an individual product and just provide semi-automated capabilities to support the generation of safety analysis assets, thus not addressing multiple SPL products.

Model-based safety analysis tools like HiP-HOPS (Hierarchically Performed Hazard Origin & Propagation Studies) [10] provide the capability to generate fault trees and FMEA analysis for a single product. Product line variability management tools like Hephaestus [11] provide a mean to establish the mapping between product line architecture design elements, hazards and their causes by means of feature expressions and transformation rules applied to these elements to derive safety-critical products. So, these tools and techniques could be combined to support the generation of fault trees and FMEA analysis for multiple SPL products, reducing the effort to perform product safety analysis.

In this paper, we propose a model-based approach to support the generation of safety analysis assets addressing multiple SPL products. The approach was used to generate fault trees and FMEA analysis for the products of an automotive hybrid braking system SPL developed based on the ISO 26262 [12] standard. Section II presents the proposed approach. Section III presents a case study conducted to evaluate the approach. Section IV presents related work. Section V presents the conclusions and future research.

II. A MODEL-BASED APPROACH TO SUPPORT MULTI-PRODUCT SAFETY ANALYSIS AND ASSESSMENT

An important requirement of safety-critical product line development processes is the inclusion of safety analysis activities such as hazard, risk, and causal analyses during product line domain engineering. These activities provide a preliminary safety assessment of the product line architecture by means of Fault Trees and FMEA assets. Such assets provide the causal information about the impact of failures in SPL design elements in the occurrence of hazards, and allocation of safety requirements necessary to minimize the hazard effects.

Fig. 1 presents the proposed approach. It was built upon the automotive domain and the HiP-HOPS tool, but it can also

be generalized and applied to other domains and model-based safety assessment (MBSA) tools. HiP-HOPS [10] is a MBSA tool that has an interface to the Simulink modeling package. HiP-HOPS provides a graphical failure editor to specify hazards to the system functions, and the failure logic (i.e. output and input deviations, and internal failures of a component that may lead to hazards) that describes how individual components can fail.

HiP-HOPS failure editor allows safety analysts to annotate the system models with the failure information in the form of extend HAZOP (HaZards & Operability) safety analysis technique [10]. An editor for annotating the system models with failure logic was built as an extension of Simulink, using its application programming interface. Once the system models have been annotated with hazards and the component failure logic, HiP-HOPS synthetizes fault trees for every system hazard in the model, and combines them to create the FMEA. HiP-HOPS presents the fault trees and FMEA analysis results in the form of hyperlinked web pages.

Product line variability management tools [2][11] were also used in the proposed approach to manage the variation in safety-critical product line design and safety analysis (i.e. *Definition of Product Line Configuration Knowledge*); and to support the *Product Derivation* according to the feature selection. Product line feature and context models are the starting point of the approach, while the main outputs are the fault trees and FMEA analysis for each SPL product. The feature model captures structural or conceptual relationships between common and variable functions of products of a domain [13]. As the approach stands on model-based development, tools like MATLAB/Simulink and HiP-HOPS can be used to support both design of the SPL architecture and safety analysis. SPL development and safety analysis activities can be performed concurrently. In the diagram of Fig. 1, `n` means the current product, and `m` represents the number of products involved in the analysis. The next subsections describe each step, its inputs and outputs.

A. Design of Product Line Architectures

In our approach, the design of a product line architecture consists of implementing the system functions specified in a product line feature model using model-based development tools like MATLAB/Simulink. The output of this activity is a set of hierarchical data-flow style models that represents the product line architecture. Simulink variability patterns available in the literature [14][15] can be used for modeling variation in SPL architectures developed with model-based development tools. Botterweck et al. [15] proposes the use of variability mechanisms of Simulink blocks to represent optional (*Enabler subsystems*), alternative (*Switch blocks*), and inclusive-or (*Integration blocks*) features.

Steiner et al. [14] have extended the Simulink variability mechanisms with two patterns to configure features with hierarchical or dependency relationships in the data flow part of Simulink models; and two patterns to configure variability in finite state machines. These patterns are also applicable to models designed in modeling environments other than Simulink. Steiner et al. [14] also proposed an approach to support variability modeling in Simulink models using variability patterns, Pure::variants, and Hephaestus variability management tools applied to an UAV-SPL. Details on how to represent variability in product line models using these patterns and tools is outside the scope of this paper.

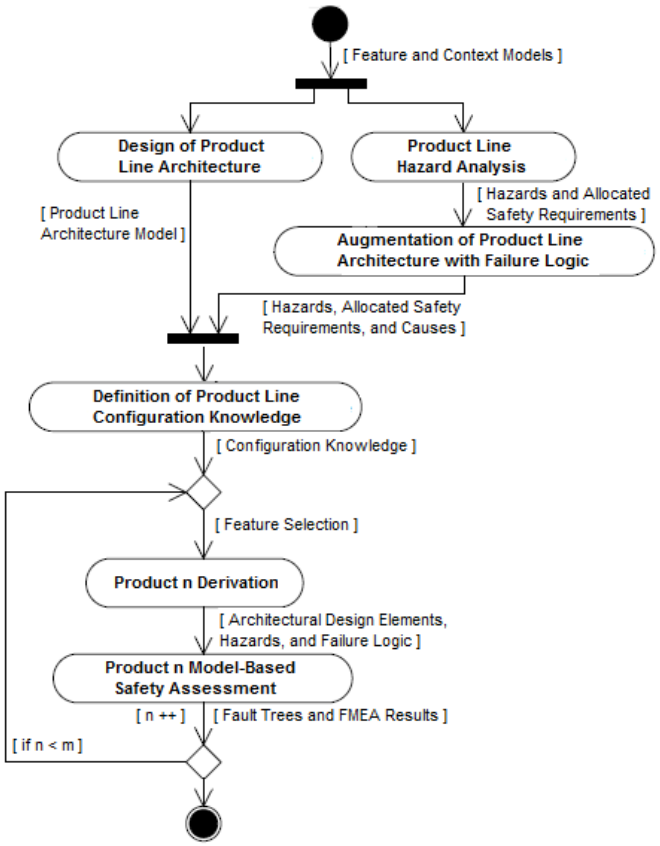

Fig. 1. Multi-product model-based safety analysis.

B. Product Line Hazard Analysis

Hazards, their causes, and the allocated safety requirements may change according to the selection of product variations. Variations in an SPL architecture are expressed in its feature model [13]. Safety requirements placed to a particular hazard may also change according to contextual elements such as operational environment, safety standards, and regulations. These elements can be represented in product line context models. For the proposal of this paper we have considered to perform the hazard analysis based only on the product line feature model. Product line features in this paper stand for system functions implemented by design elements (e.g. system, subsystems, components).

The product line hazard analysis aims to identify the possible hazards that can arise from failures in common and variable design elements and to allocate requirements to minimize the effects of these hazards. As it would be prohibitive performing a hazard analysis covering all possible product variants, we suggest constraining the scope of the analysis to a set of clearly defined products. Products can be defined by deriving instantiation scenarios from the analysis of the product line feature model. The criteria to constrain the scope of the analysis can be the SPL variations that are most relevant for the stakeholders.

After scoping the analysis to a set of common and variation points, model-based hazard analysis can be performed, for example using the extended HAZOP analysis technique supported by HiP-HOPS tool [10]. Hazards and safety requirements can be specified via the HiP-HOPS failure editor. HiP-HOPS hazard analysis can be performed in parallel with the SPL architecture design. In HiP-HOPS, hazards are specified by means of logical expressions involving possible failures in design elements. These failures are generally stated in terms of failure types that typically include omission, commission, value, early, and late failures. Product line hazard analysis using HiP-HOPS can be performed by identifying the hazards associated to system functions common to all products; and by identifying the hazards associated to system functions standing for SPL variation. Next, safety requirements are allocated to the identified hazards. HiP-HOPS stores the hazard analysis data in a failure model.

C. Augmentation of Product Line Architecture with Failure Logic

Variation in product line architecture may change the causes of a particular hazard. Describing how product line architectural design elements (i.e. product line components) can fail and contribute to the occurrence of each hazard identified in *Product Line Hazard Analysis* is the main goal of this step. We should specify the failure data inherent to each SPL design element, by stating what can go wrong with such element and how it responds to failures elsewhere in the architecture. Such information is called component failure logic. HiP-HOPS allows to annotate the product line architecture with a set of failure expressions showing how deviations in component outputs can be caused either by internal failures in the component or corresponding deviations in component inputs.

Deviations may include unexpected omission of an output or unintended commission of output, or incorrect outputs values, or the output being too early or late [10]. HiP-HOPS failure editor also provides a graphical user interface to specify the failure logic of each individual product line component. HiP-HOPS stores such failure logic in a library, so that other components of the same type can reuse the failure logic. The failure logic inherent to each product line design element is stored by HiP-HOPS into the product line failure model together with the hazard analysis information.

It is important to state that this local failure analysis can reflect either real characteristics or simply the design intention for the analyzed design elements. In both cases the analysis is useful. For example, at early stages when components are under design and only design intentions are encoded, it is still possible, using model-based safety analysis, to assess the suitability of the proposed design under these encoded intentions about failure logic, fault propagation, fault mitigation and fault tolerance of various design elements. Such analysis can help to identify weaknesses and decide how to improve the design e.g. by introducing elements with improved characteristics or fault tolerant features. Qualitative analysis is only needed for this purpose, while difficult to obtain or unavailable failure rates for software elements are not necessary.

D. Definition of Product Line Configuration Knowledge

Up to this point, we have identified the product line variabilities during the design of the SPL architecture (II-A subsection) by means of variation mechanisms such as *Switches* and *Enabler Subsystems*. We also have specified the variations in product line safety analysis by assigning hazards and placing requirements to common and variable SPL design elements (II-B subsection); and by specifying the failure logic inherent to each SPL design element (II-C subsection). In this step, the SPL configuration knowledge is established through a set of rules to manage the variation in both design and safety analysis assets. Such rules describe how these assets can be composed in a product according to the feature selection.

The main information required to specify the product line configuration knowledge is derived from the SPL feature model. Product line variability management tools can be used to support the configuration knowledge definition. Tools like Pure::variants [2] provide support for negative variability, i.e. product derivation is based on the activation and deactivation of design elements according to the feature selection. Model transformation tools like Hephaestus/Simulink [14] support positive variability, i.e. the product derivation includes only the SPL design elements that correspond to the features specified in the feature selection.

Product line configuration knowledge can be specified using model transformations tools like Hephaestus/Simulink [14] by applying the following steps: 1) specify the feature expressions in the scope of the usage scenarios considered in the hazard analysis. A feature expression may include a single feature or a combination between two or more features; 2) for each feature expression, determine the product line design elements to be included and excluded; and 3) specify the hazards, the allocated safety requirements, and the failure logic to be included/excluded in each feature expression. After performing these steps, we obtain the mapping between product line features, design elements, hazards and the allocated safety requirements, and component failure logic. Additional details on how to use product line variability management tools to specify the configuration knowledge can be found elsewhere [14].

E. Product Derivation

After establishing the rules to compose design and safety analysis assets (i.e. hazard, and failure logic), products can be derived. Variability management tools used for specifying the configuration knowledge can also be used to provide the automated support for product derivation according to the specified feature selection. The output of product derivation is the product architecture model, hazards and allocated requirements, and the failure logic (i.e. HiP-HOPS product failure model) corresponding to the feature selection of a particular product. This information will be further used to perform automatic fault trees and FMEA analysis for a particular product.

F. Product Model-Based Safety Assessment

Product line safety analysis activities may reduce the effort in performing product safety analysis and assessment by providing not all, but a set of product-specific hazards and

failure logic. After the product derivation, HiP-HOPS can be used to perform product safety analysis by adding product-specific hazards, and failure logic to the product failure model. Next, HiP-HOPS can be used for generating product-specific fault trees and FMEA results from the product architecture model and its HiP-HOPS failure model. HiP-HOPS stores product-specific fault trees and FMEA analysis results in a "fault trees" XML file, and presents it in the form of hyperlinked HTML pages. `n` is incremented at the end of the loop until it reaches `m` (i.e. the number of products involved in the analysis). The accuracy of product-specific HiP-HOPS fault trees and FMEA analysis depends on whether product line safety analysis activities were performed aware of safety-related variation.

III. Case Study

The Hybrid Braking System [16] automotive product line (HBS-SPL) was chosen to evaluate the proposed model-based approach to support automatic safety analysis of multiple SPL products. Three different HBS-SPL products were considered in this case study. The results of applying the proposed approach to HBS-SPL products were used as a proof of concept.

A. Hybrid Braking System Product Line

HBS-SPL is a real world automotive braking system product line designed in MATLAB/Simulink. HBS-SPL is meant for electrical vehicles integration, in particular for propulsion architectures that integrate one electrical motor per wheel [16]. The term *hybrid* comes from the fact that braking is achieved throughout the combined action of the electrical In-Wheel Motors (IWMs) and frictional Electromechanical Brakes (EMBs). One of the most important features of this system is that the integration of IWM in the braking process allows an increase in the vehicle's range: while braking, IWMs work as generators and transform the vehicles kinetic energy into electrical energy that is fed into the powertrain battery.

HBS-SPL components can be combined in different ways according to the constraints specified in HBS-SPL feature model presented in Fig. 2. It includes wheel braking alternative features: Brake_Unit1_Front, Brake_Unit2_Front, Brake_Unit3_Rear, and Brake_Unit4_ Rear aimed to provide the braking for each wheel; Mechanical Pedal, a hardware device aimed to capture driver presses; Electronic Pedal, a hardware device that senses and processes the actions from the mechanical pedal; Bus1 and Bus2 features send the wheel braking forces to the wheel braking units; Auxiliary Battery feature, a hardware device responsible for feeding the electromechanical brakes while braking; and Powertrain Battery, a hardware device that receives the electrical energy produced by the in-wheel motors. HBS-SPL feature model also presents the constraints showing how features can be composed in a product.

B. HBS Product Line Hazard Analysis

Hazards can arise from the interaction between HBS-SPL design assets in a range of usage scenarios. Different hazards can arise according to how HBS-SPL design elements can be composed in a product. Performing a hazard analysis covering all possible usage scenarios for HBS-SPL design elements would be prohibitive. Nevertheless, scoping the HBS-SPL hazard analysis to a set of products brought some degree of reuse for safety analysis assets. The extended HAZOP analysis technique [10] and HiP-HOPS were used to perform the HBS-SPL hazard analysis. Wheel Braking variation point specified in the HBS-SPL feature model was considered in the hazard analysis. From the analysis of Wheel Braking variation point and mandatory elements of HBS-SPL, the following usage scenarios were established: HBS four wheels braking (HBS-4WB); HBS front wheels braking (HBS-FWB); and HBS rear wheels braking (HBS-RWB). These scenarios were analyzed from the safety perspective. Table I presents the identified hazards, their causes, and the allocated ASILs (Automotive Safety Integrity Levels) to minimize their effects. Table I also presents the association between the hazards and the usage scenarios by means of the column "Usage Scenario".

In order to simplify the case study, we have not assigned ASILs to product line hazards on the basis of the full ISO 26262 [12] risk assessment process. We have derived such ASILs only considering the hazard severity. In a product line hazard analysis, different ASILs can be assigned to the same hazard considering different usage scenarios for product line design elements. For example, the ASIL allocated to the "No braking rear" hazard is more stringent in HBS-RWB scenario

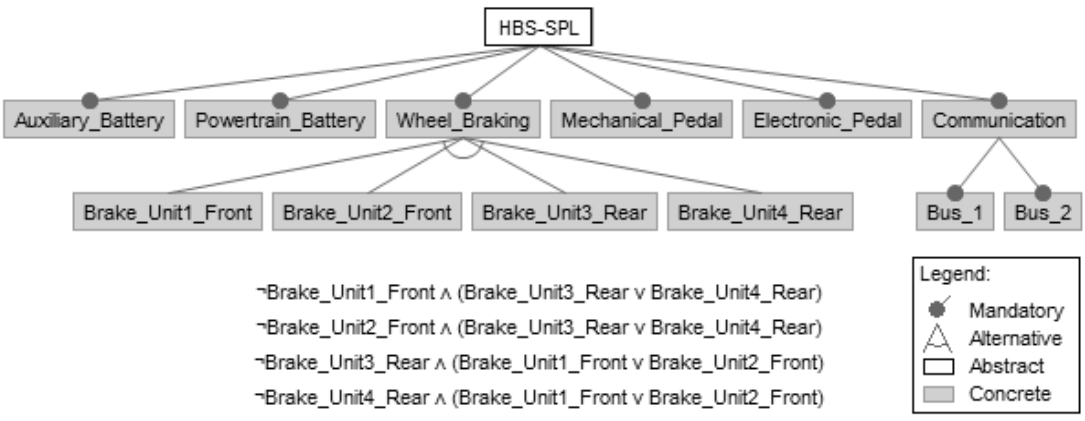

Fig. 2. HBS-SPL feature model.

TABLE I. HBS PRODUCT LINE HAZARDS AND ASILS.

Usage Scenario	Hazard	Causes	ASIL
HBS-4WB	No braking four wheels	Omission of all brake unit actuators outputs.	D
	No braking rear	Omission of brake unit3 and brake unit4 actuators outputs.	C
	Value braking	Incorrect value of all brake unit actuators outputs	C
HBS-FWB	No braking front	Omission of brake unit1 and brake unit2 actuators outputs.	D
	Value braking	Incorrect value of brake unit1 and brake unit2 actuators outputs.	D
HBS-RWB	No braking rear	Omission of brake unit3 and brake unit4 actuators outputs.	D
	Value braking	Incorrect Value of brake unit3 and brake unit4 actuators outputs.	D

and less stringent in HBS-4WB. Causes for a particular hazard can also change according to how product line design elements can be composed in a product. The causes for the "Value braking" hazard in HBS-FWB are different from the causes for that hazard in HBS-RWB. HBS-SPL hazards and ASIL allocation are stored by HiP-HOPS in the failure model.

C. Augmentation of HBS Product Line Architecture with Failure Logic

From product line hazards, 77 failure logic expressions inherent to 30 HBS-SPL design elements were added to the product line failure model via HiP-HOPS failure editor. Failure logic of a design component is stated in HiP-HOPS by specifying output deviations and the possible input and internal failures that may lead to the output deviations. Table II presents the failure logic for the Wheel Node Controller (WNC) component. Omission, commission, value, early, and late guidewords were used to express the failure logic of HBS-SPL components. Input and internal failures contributing to the occurrence of output deviations were specified by means of failure expressions. For example, the omission of the WNC output can be caused by an internal failure in WNC or the omission of its inputs.

TABLE II. WNC COMPONENT FAILURE LOGIC.

Component	Output Deviation	Failure Expression
WNC	Omission-Out1	WNCOFailure1 or (Omission-In1 and Omission-In2)
	Value-Out1	WNCVFailure1 or Value-In1 or (Omission-In1 and Value-In2)

D. Definition of the Configuration Knowledge

The Hephaestus/Simulink [11][14] variability management tool was used to specify the HBS-SPL configuration knowledge. HBS-SPL design elements (i.e. Simulink model components), hazards, and component failure logic were mapped to feature expressions in Hephaestus. Feature expressions were defined from the HBS-SPL feature model. Transformation rules were also established to determine how HBS-SPL components, hazards, and component failure logic can be composed in a SPL product.

E. HBS Product Derivation

Hephaestus/Simulink [11][14] was also used to support the automated derivation of the following HBS-SPL products: HBS-4WB, HBS-FWB, and HBS-RWB. HBS-4WB architecture includes mechanical pedal and electronic pedal that sends the braking outputs to the communications buses. Communication buses send the braking commands to Brake_Unit1_Front, Brake_Unit2_Front, Brake_Unit3_Rear, and Brake_Unit4_Rear model components. HBS-FWB product differs from HBS-4WB by the absence of rear wheels braking units. HBS-RWB product differs from HBS-4WB by the absence of front wheels brake units. Hazards, causes, and components are different in each one of these products. For each product, the hazard analysis data is stored together with the product Simulink model in the product failure model.

F. HiP-HOPS Model-Based Safety Assessment

After the derivation of HBS-SPL products and their failure models using Hephaestus, each product was used as input to HiP-HOPS [10] to perform the safety assessment. From HiP-HOPS analysis, fault trees, failure cut sets, and FMEA results were generated for each HBS-SPL product. Fault trees were generated for each product-specific hazard described on Table I. HiP-HOPS FMEA results describe the relationships between direct and further effects of a hazard and the failure modes. From the analysis of HBS-SPL products FMEA results, we have identified 9 single-point and 39 multi-point failure modes for HBS-4WB, 5 single-point and 33 multi-point failure modes for HBS-RWB, 5 single-point and 33 multi-point failure modes for HBS-FWB. This analysis has given insights into the design of the product line, for instance common single points of failure that affect different products and contribute to significant hazards that arise across the SPL, but also more subtle findings about the failures of individual SPL products.

An example of the above is the failure mode "*Omission of Brake_Unit3_Rear and Omission of Brake_Unit4_Rear*" of the Wheel Braking component which causes "No braking rear" hazard in HBS-4WB and HBS-RWB products. Unfortunately, due to space limitation it is not possible to present in detail the results of the generated analysis. We should note that HiP-HOPS can also be used for allocating system safety requirements to elements of a design. So, from the ASILs allocated to HBS-SPL hazards, it is possible to automatically allocate ASILs to design elements, and thereby determine the safety requirements for those elements that must be met to fulfill the safety requirements of the product line [10]. This is indeed the subject of current research that extends the proposed framework for MBSA of product lines.

IV. RELATED WORK

Research in product line safety analysis has focused on adapting traditional safety analysis techniques, such as FTA and FMEA to suit product line processes. The most notable work addressed to this topic is the extension of Software FTA

(SFTA) to address the impact of SPL variation on safety analysis [6][7][8]. This approach is based on a technique for the development of a product line SFTA in the domain engineering phase, and a pruning technique to reuse such SFTA for the analysis of SPL instances. It offers a systematic approach to treat SFTA results as a reusable asset. Such approach was later extended to integrate product line SFTA analysis results with model-based development using state-based models [9]. However, the product line SFTA approach provides a semi-automated generation of SPL fault trees that requires support of domain expert reviews. It does not support the generation of FTA that reflects the complexity of product line features with context (product variants). SFTA approach is purely hierarchical and not sensible to the different SPL products. The novelty of our approach is the provision of guidelines to reuse product-specific hazards and their causal information, and to use the HiP-HOPS tool to generate product-specific fault trees and FMEA analysis from the product hazards and causes. Different from [6][7][8] [9], the proposed approach is sensible to product variants and does not require support of domain expert reviews. In our approach, the domain expert knowledge is stored in the configuration knowledge. Variability management tools use the configuration knowledge to provide automated consistency verification of the composition of product line design and safety analysis assets in a product.

Blessing and Huhn [17] proposed a model-based formal safety analysis approach addressing a Pacemaker product line. The approach uses Common Variability Language (CVL) [18] for variability modeling, and the SCADE suite for system modeling. In their approach, SCADE automated model checking was used to prove the validity of safety requirements of Pacemaker variants. Their formal safety analysis approach is focused in automatic derivation of Pacemaker products using CVL, and in the formal verification of product safety requirements. The approach proposed in this paper uses variability management tools like Hephaestus for product derivation, and it is focused in automated generation of fault trees and FMEA addressing multiple SPL products using HiP-HOPS tool. CVL could be a possible way to manage the variation in SPL design and safety analysis in our approach.

V. CONCLUSION

Safety is highly connected to the system, so it should be considered early in domain engineering, when establishing the SPL architecture. Achieving the reuse of SPL safety analysis assets requires performing the safety analysis aware of interactions between SPL design assets in possible usage scenarios. The novelty of the approach proposed in this paper is the provision of guidelines prescribing how model-based development, safety analysis, and variability management tools can be used to reduce the effort of product safety analysis. These guidelines provide a mean to trace product line variation throughout SPL design and safety analysis. The proposed approach is tool independent, in which MBSA tools like HiP-HOPS can be used to automatically generate fault trees and FMEA results for multiple SPL products. Thus, the results obtained from this study can be generalized to other systems from different domains like avionics.

Fault trees and FMEA assets can be used for structuring the product safety case argument organized into multi-view point argument modules [5] addressing product-specific hazards and allocated requirements, and the contribution of component failures to hazards. As future work, we propose a safety case pattern for SPL product argumentation using Goal Structuring Notation (GSN) [19]. We also intend to implement a tool to support the generation of safety cases, using such pattern, addressing multiple SPL products.

ACKNOWLEDGMENT

Our thanks to CNPq, process number 152693/2011-4.

References

[1] I., Crnkovic, “Component-based software engineering for embedded systems” Proceedings of the 27th International Conference on Software Engineering, St. Louis, MO, EUA, ACM, New York, 2005, p. 712-713.

[2] J., Weiland, “Configuring variant-rich automotive software architecture models”, 2nd IEEE Conf. on Automotive Electronics, 2006, pp. 73-80.

[3] F., Dordowsky, R., Bridges, H., Tschope, “Implementing a Software Product Line for a Complex Avionics System”, Proc. of the 15th Int. Software Product Line Conference, IEEE, 2011, p. 241-250.

[4] P., Clements, L., Northrop, Software Product Lines: Practices and Patterns. Addison-Wesley, 2001.

[5] I., Habli, and T., Kelly, “A safety case approach to assuring configurable architectures of safety-critical product lines”, Proc. of the 1st Int. Conf. on Architecting Critical Systems, Springer-Verlag, 2010, pp. 142-160.

[6] J., Dehlinger, R., Lutz, “PLFaultCAT: A Product line Software Fault Tree Analysis Tool”, Automated Software Engineering, vol. 13, n. 1, pp. 169-193, 2006.

[7] J., Dehlinger, R., Lutz, “Software Fault Tree Analysis for Product Lines”, Proc. of the 8th IEEE International Symposium on High Assurance Systems Engineering, Florida, USA, 2004.

[8] Q., Feng, R., Lutz, “Bi-Directional Safety Analysis of Product Lines”, Journal of Systems and Software, vol. 78, n. 2, pp. 111-127, 2005.

[9] J., Liu, J., Dehlinger, R., Lutz, “Safety Analysis of Software Product lines Using Stated Modeling”, Journal of Systems and Software, vol. 80, n. 11, pp. 1879-1892, 2007.

[10] L., Azevedo, D., Parker, M., Walker, Y., Papadopoulos, R., Araujo, "Assisted Assignment of Automotive Safety Requirements". IEEE Software 31(1):62-68, 2014, IEEE.

[11] R., Bonifácio, L., Teixeira, P., Borba, “Hephaestus: A tool for managing product line variabilities”, 3rd Brazilian Symposium on Components, Architecture, and Software Reuse, pp. 26-34, 2009).

[12] ISO, ISO 26262: Road Vehicles Functional Safety, 2011.

[13] K., Lee, K. C., Kang, J., Lee, J., “Concepts and Guidelines of Feature Modeling for Product Line Software Engineering”, Proc. of the 7th Int. Conf. on Software Reuse: Methods, Techniques, and Tools, Springer-Verlag, London, UK, 62-77, 2002.

[14] E. M., Steiner, P. C., Masiero, “Managing SPL Variabilities in UAV Simulink Models with Pure::variants and Hephaestus”, CLEI Electronic Journal, v. 16, n. 1, 2013.

[15] G., Botterweck, A., Polzer, S., Kowalewski, “Using higher-order transformations to derive variability mechanism for embedded systems" Models in Software Engineering, pp. 68-82, 2010.

[16] R., De Castro, R. E., Araújo, D., Freitas, “Hybrid ABS with Electric motor and friction Brakes’, 22nd International Symposium on Dynamics of Vehicles on Roads and Tracks, Manchester, UK, 2011.

[17] S., Blessing, M., Huhn, "Formal Safety Analysis and Verification in the Model Driven Development of a Pacemaker Product Line", MBEES, 2012.

[18] O., Haugen, B., Moller-Pedersen, J., Oldevik, G. K., Olsen, A., Svendsen, "Adding Standardized Variability to Domain Specific Languages," 12th International Software Product Line Conference, pp.139,148, 2008.

[19] T., Kelly, “A systematic approach to safety case management”, SAE world congress, Society for Automotive Engineers, 2003.