Supporting the Automated Generation of Modular Product Line Safety Cases

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, Brasil,
{andre_luiz, rtvb, masiero}@icmc.usp.br
[2]Department of Computer Science, University of Hull, Hull, United Kingdom
y.i.papadopoulos@hull.ac.uk
[3]Department of Computer Science, University of York, York, United Kingdom
{ibrahim.habli, tim.kelly}@york.ac.uk

Abstract. The effective reuse of design assets in safety-critical Software Product Lines (SPL) would require the reuse of safety analyses of those assets in the variant contexts of certification of products derived from the SPL. This in turn requires the traceability of SPL variation across design, including variation in safety analysis and safety cases. In this paper, we propose a method and tool to support the automatic generation of modular SPL safety case architectures from the information provided by SPL feature modeling and model-based safety analysis. The Goal Structuring Notation (GSN) safety case modeling notation and its modular extensions supported by the D-Case Editor were used to implement the method in an automated tool support. The tool was used to generate a modular safety case for an automotive Hybrid Braking System SPL.

Keywords: Product lines, certification, modular safety cases, reuse.

1 Introduction

Safety cases are required by certification bodies for developing automotive, avionics, and railway systems. Safety standards like ISO 26262 [11] and EUROCAE ED-79A [6] require the use of safety cases. Software product lines (SPL) have been used in the development of automotive [12] and avionics [8] systems, reducing the development effort. Ideally, safety analysis and safety cases should be treated as reusable assets in safety-critical SPLs. Such reuse of pre-certified elements would reduce re-certification effort contributing to modular and incremental certification. We refer to that concept as modular certification and we argue that it can provide means of dealing effectively with the impact of changes caused by variation of features in products within a SPL. Note that modular certification has already been shown effective in handling changes in the certification of a system as it is updated through its life [7].

Research in SPL provided a number of systematic approaches for managing the impact of variations in changing the system architecture by means of feature and context models [3]. A feature model captures the main system functions offered for reuse

in a SPL. A context model captures information about the physical, operating, support, maintenance, and regulatory environment of SPL products [9].

Clearly, each product within a safety-critical SPL would need a safety case that reflects its specifics characteristics. For such safety cases to be produced efficiently, variation must be traced transparently in feature and context models across SPL design and safety analyses [9][10]. Information like hazards, their causes, and the allocated safety requirements (derived from the hazard analysis) may change according to the selection of SPL variation in feature and context models in a particular product [9][10][12]. Such variation may also change the structure of the product safety case. SPL variability management tools and techniques like Hephaestus [18] and Pure::variants [12] can be used to manage feature and context variations in SPL design and safety analysis assets by means of definition of the configuration knowledge.

There exist numerous methodologies and tools for semi-automated model-based safety analysis that can be adapted for SPL design. One of these tools, HiP-HOPS (Hierarchically Performed Hazard Origin & Propagation Studies) [17] provides Fault Trees Analysis (FTA), Failure Modes and Effect Analysis (FMEA), and safety requirements allocation information that are mostly auto-generated and could potentially be used to inform a safety case. However, the direct translation of such information to safety arguments would very likely result in monolithic safety arguments which may lack structure and not help to identify the arguments and evidence for specific elements of the system architecture. So, while it would be useful to exploit the safety analysis assets provided by model-based safety analysis, one would need to find a 'clever' way to argue the internal stages and nuances of the argument.

In order to support the reuse of safety arguments, a modular SPL safety case architecture could be built in a way that reflects the typical system decomposition into more refined subsystems [7][12]. Thus, it would be possible to identify the properties (i.e. goals, sub-goals, and evidence) required by individual safety case modules. The establishment of a modular safety case requires the identification of boundaries (i.e. stopping points) in the system architecture to guide the rationale of decomposing the safety case into argument modules [7]. Feature and context models can provide such stopping points. These boundaries can be used to establish the structure of the argument modules and interdependences between them according to the following well-established software architecture principles: High Cohesion/Low Coupling, Divide and Conquer, Extensibility, and Isolation of Change (i.e. Maintainability) [12].

GSN [16] and CAE (Claim, Argument, Evidence) [2] are two notations largely adopted by industry and are both applicable for modular safety cases. GSN patterns [13] and modular extensions [10] support the development of reusable modular safety case architectures. Existing safety case modeling tools such as D-Case Editor [15] and ACEdit [1] support GSN and its modular extensions and provide an Eclipse-based graphical editor. Particularly, D-Case implements a formal definition of safety case modeling language based on GSN and its extensions.

Although earlier work has partly addressed SPL safety assessment [9] and safety case development [10], there is still a need to automate the traceability of SPL variation starting from architectural models to support the generation of a modular safety case architectures addressing product lines. In this paper we propose a novel approach

supporting largely automated generation of modular and reusable safety arguments from system models and safety analyses. Section 2 presents the proposed method to support the generation of modular safety cases and its instantiation in a tool support, Section 3 discusses a case study and Section 4 presents the conclusion.

2 The Proposed Method

The production of a modular SPL safety case requires the traceability of SPL feature and context variation across architecture and safety analyses. Habli's SPL safety metamodel [9] can support such traceability. It includes abstractions to represent SPL variation in feature and context assets and their impact on the hazards, their causes, and the allocated requirements. The metamodel also describes the relationships between SPL variation and assurance assets (e.g. safety argument elements in GSN). The provision of automated support for this metamodel can facilitate the automatic generation of modular product line safety cases. In the following we present a method to create support for automatic generation of modular safety cases from a number of inputs which include: the SPL safety metamodel [9], the safety case modeling notation metamodel (e.g., D-Case [15] GSN metamodel) and safety case patterns [13][19].

The safety metamodel specifies core and variant system and contextual assets. It also captures the association between functional, architectural, and component-level failures with specific SPL assets specified in the product line context, feature and reference architecture models. These relationships are captured in a Functional Failure Model (FFM), an Architectural Failure Model (AFM), and a Component Failure Model (CFM). The three models support explicit traceability between development and safety assessment assets. The FFM captures the impact of contextual and functional (i.e. feature) variation on the SPL functional failure conditions, their effects, and allocated safety requirements. The AFM records the impact of contextual and design variation on the SPL architectural failure modes and safety requirements. The AFM also produces the failure mode causal chain and derived safety requirements contributing to the functional failure conditions previously captured in the FFM. Finally, the CFM records the impact of design variation on component failure modes, their effects, and means of mitigation (i.e. specific safety requirements). Details of these models can be found in [9]. In our approach, model-based development, safety analysis, and SPL variability management tools provide the information required to instantiate the SPL safety assessment metamodels. The modeling and analysis infrastructure of such tools, together with the D-Case GSN metamodel, and safety case patterns [13][19] are then used to auto-generate SPL modular safety cases.

Fig. 1 presents the structure of the proposed method in an UML activity diagram. Functional Failure Modeling requires the feature/instance model (i.e. FM or IM), and the system model (SMDL) as inputs to produce instances of FFMs. The Design of Modular Safety Case step requires the following inputs: FFMs, the SPL feature model (FM), Safety Case Modeling Notation and its metamodel to produce the structure of the safety case architecture. Architectural Failure Modeling requires the FFMs produced in the earlier step, fault trees (FTA) and requirements allocation (RAL) infor

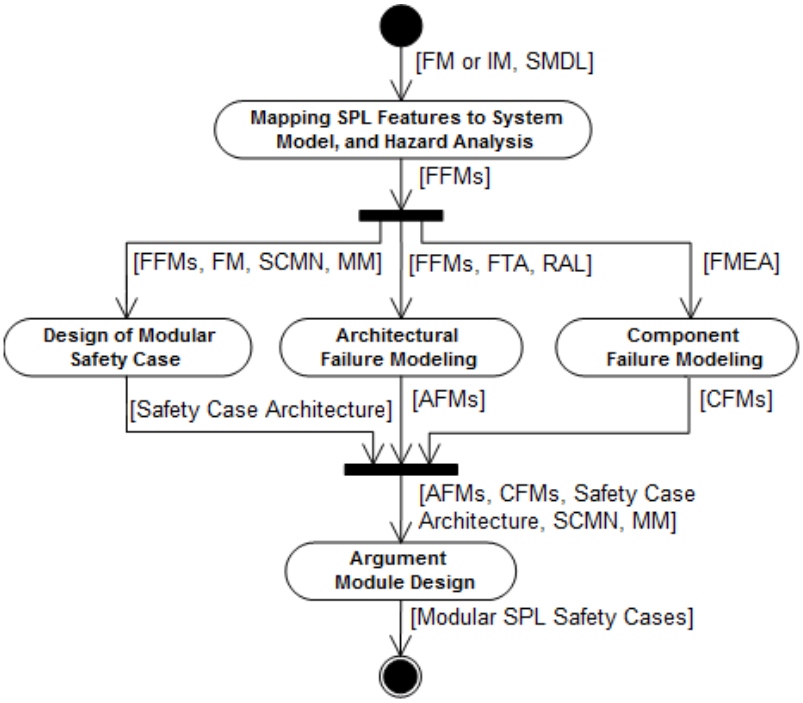

Fig. 1. Method to support automated construction of modular product line safety cases.

mation provided by model-based safety analysis to generate instances of AFMs. Component Failure Modeling requires FMEA information provided by model-based safety analysis as input, producing instances of CFMs as outputs. Finally, the Argument Module Design step generates the internal details of each argument module of the safety case architecture from the information provided by AFMs, CFMs, safety case architecture, Safety Case Modeling Notation and its metamodel.

A tool architecture was developed to support the method presented in Fig. 1 and it is illustrated on Fig. 2. Certain SPL assets provide inputs in this architecture instantiating FFM, AFM, and CFM models: the system model annotated with safety/hazard analysis data provided by the MATLAB/Simulink and HiP-HOPS (i.e. SMDL in Fig. 2); the feature model (FM), the instance model (IM), and configuration knowledge (CK) provided by the Hephaestus/Simulink variability management tool; and the fault trees (FTA), FMEA, and requirements allocation (i.e., RAL) generated by HiP-HOPS. The D-Case GSN metamodel and safety case patterns were used to organize the structure of the modular safety case and product-based arguments for each SPL feature and its variation. It is important to highlight that other tools can be used to develop other instances of a similar tool support. The method presented in Fig. 1 is general and can be applied manually or with the use of a set of different tools. The specific set of tools used in this paper is discussed below.

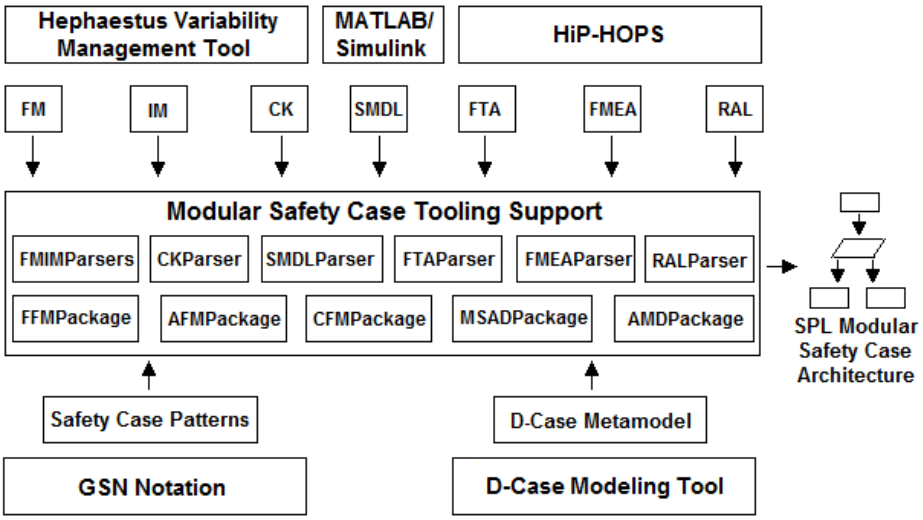

Fig. 2. Modular product line safety case tooling support.

2.1 Functional, Architectural, and Component Failure Modeling

A product line variability management tool, in this case Hephaestus/Simulink [18], provides the specification of the SPL feature model (or instance model in product derivation), and the configuration knowledge, i.e., it describes the mapping between SPL features, and the design elements of the system model by associating component identifiers with feature expressions. A model-based development environment, in this case MATLAB/Simulink, provides the specification of the design elements of the system architecture. HiP-HOPS [17] was used in this work to annotate the system model with hazard/safety analysis information describing the system hazards, their causes, allocated safety requirements, and the failure logic of each design element. To obtain the necessary information for mapping SPL features, system model elements, and hazard/safety analysis data in the tool support, parsers were implemented for the SPL feature model (FMIMParsers), and configuration knowledge (CKParser) using the Java XStream[1] XML library. A Simulink parser for Java[2] was used to obtain information about the structure of the system model and its safety annotations. From the information provided by these parsers, the mapping between SPL features, design elements, and hazards is obtained by instantiating FFMs. This information is further used to provide instances for the SPL functional failure model (FFMPackage).

The HiP-HOPS tool provides fault trees, FMEA analysis, and allocation of safety requirements to architectural modules in the form of Safety Integrity Levels (in this case Automotive SILs or ASILs as our case study in an automotive system) in a XML file. This file contains the information about the failure mode causal chain of each SPL hazard, and the requirements allocated to each failure mode involved in the caus-

[1]XStream: http://xstream.codehaus.org/

[2]Simulink Library for Java: https://www.cqse.eu/en/products/simulink-library-for-java/overview/

al chain. The mapping between features, system model elements, and system hazards obtained in FFM is used together with HiP-HOPS analyses to instantiate the product line AFM. Parsers were developed to manipulate the fault trees (FTAParser) and requirements allocation (RALParser) generated by HiP-HOPS. These parsers are used together with the mapping between features and Simulink model elements to instantiate the SPL architectural failure model (AFMPackage).

CFM complements the product line AFM with a component-level view of the failure behavior of the system. CFM describes the local effects of a component failure mode, i.e. effects occurring at the boundary of the component, typically its outputs. FMEA generated by tools like HiP-HOPS provides the information required to instantiate the CFM. The developed tool reads this information (via FMEAParser) and instantiates the CFM elements (CFMPackage).

2.2 Design of Modular Safety Case Architecture

We describe how to represent the SPL variation expressed in the feature model using modular GSN [16], and how to use the D-Case tool GSN metamodel to design the architecture of a modular safety case. Czarnecki et al. [3] have provided abstractions to represent relationships between features such as inclusive-or (1-n), exclusive-or (1-1) features, and exclusive-or with sub-features (0-n). Fig. 3 shows how to specify variation expressed in the feature model into the design of a GSN modular safety case. Argument modules arguing the safety of mandatory features (Fig. 3a) is expressed by a top-level argument module connected to a module *'A'* via the *'supported by'* relationship (Fig. 3b). The multiplicity of such arguments capturing multiple occurrences of a mandatory feature can be represented by means of the *'supported by'* relationship with a filled circle (GSN multiplicity). An argument module arguing the safety of an SPL optional feature (Fig. 3c) is represented in GSN by connecting the top-level argument to the optional module *'B'* using the '*supported by'* relationship with an empty circle (Fig. 3d). Argument modules arguing the safety of SPL features involved in inclusive-or (at least 1), exclusive-or (0 or n), or exclusive-or (1 of n) with grouping relationships (e.g., Fig. 3e) is encapsulated in GSN contracts (Fig 3f). The GSN optionality extension (i.e. choice) is used to connect the contract module to each argument module involved in a grouping relationship. Safety case modeling tools such as the Eclipse-based D-Case Editor [15] and ACEdit [1] provide implementations for the GSN metamodel, covering its patterns and modular extensions [16]. In the implementation of the method in the tool presented on Fig. 2, the D-Case GSN metamodel (i.e., Safety Case Modeling Notation and Metamodel) was used for developing the mechanism to automatically generate a modular safety case architecture. Initially, the tool (i.e., MSCDPackage) generates a top-level argument for the root feature, i.e., the feature representing the product line. Next, argument modules are created for each mandatory, optional, and grouping of features. The heuristics presented in Fig. 3 have also been applied to generate the structure of the modular safety case. Thus, an argument module has been created for the root feature of a group. This module is further connected to a GSN contract that encapsulates optional and alternative argument modules from the root argument module. At the end, a modular safety case reflecting the product line feature model decomposition is obtained.

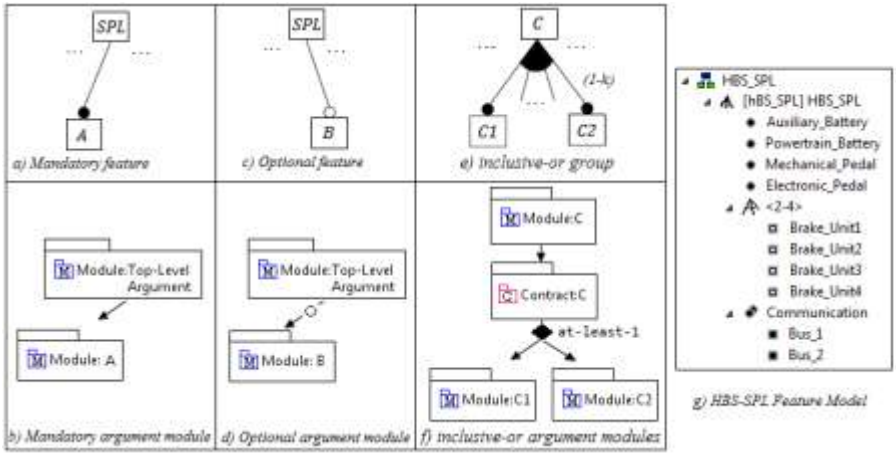

Fig. 3. Variability modeling in modular GSN and HBS-SPL feature model.

2.3 Argument Module Design

This step delivers the internal structure of each argument module arguing the safety of SPL features and their variants. The mapping between SPL features, architecture, component and safety analysis data provided by functional, architectural, and component failure models is used to auto-generate the claims that compose feature-specific safety arguments. All these information together with the D-Case GSN metamodel are used as input for implementing the mechanism to auto-generate the internal structure of feature-specific argument modules in the tool support (i.e., AMDPackage). The Hazard Avoidance pattern [13] and Weaver's 'Component Contributions to System Hazards' pattern catalogue [19] have also been used to structure each feature-specific argument. Specifically, the Hazard Avoidance pattern is used to decompose the argument over the mitigation of system hazards associated to a particular feature. The Component Contributions to System Hazards pattern catalogue is used to decompose the arguments over the mitigation of the system hazards into arguments arguing the mitigation of each component failure mode contributing to the failure of a feature.

The relationships between SPL features, hazards, and allocated safety requirements provided by the functional failure model were used for arguing the mitigation of the system hazards associated with each SPL feature. The relationships between failure mode causal chains and design assets captured in the AFM, and the relationships between components, failure modes, and effects captured in the CFM have been used for arguing the mitigation of each component failure mode associated to a particular feature.

3 Case Study

The method and tool developed in this work was used for auto-generating a modular safety case architecture for a Hybrid Braking System automotive product line (HBS-SPL). The HBS-SPL is a prototype automotive braking system SPL designed in

MATLAB/Simulink. HBS-SPL is meant for electrical vehicles integration, in particular for propulsion architectures that integrate one electrical motor per wheel [5]. 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 key 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. IWMs have, however, braking torque availability limitations at high wheel speeds or when the powertrain battery is close to full state of charge. EMBs provide the torque needed to match the total braking demand. HBS-SPL components can be combined in different ways according to the constraints specified in the HBS-SPL feature model presented in Fig. 3g. The feature model was designed using the cardinality-based notation [3] and the FMP[3] modeling tool. The HBS-SPL feature model includes wheel braking alternative features: Brake_Unit1 and Brake_Unit2 front wheels braking, Brake_Unit3 and Brake_Unit4 rear wheels braking aimed to provide the braking for each wheel. The Hephaestus/Simulink variability management tool was used to manage the variation in HBS-SPL design and safety analysis assets. Thus, the mapping between SPL features and these assets has been specified in the configuration knowledge.

3.1 HBS-SPL Hazard Analysis

SPL hazard analysis was performed in HiP-HOPS from the perspective of the following product configurations: HBS four wheel braking (i.e. all product line presented in Figure 3g) (HBS-4WB); HBS front wheel braking (HBS-FWB), which includes Brake_Unit1 and Brake_Unit2 features; and HBS rear wheel braking (HBS-RWB) that includes Brake_Unit3 and Brake_Unit4 features. Table 1 presents the identified hazards for each SPL usage scenario and their allocated ASILs. Some of these hazards are common between all HBS-SPL products and some of them are product-specific (e.g., 'No braking four wheels'). The 'Value braking' hazard is common across HBS-SPL products, but its causes and allocated ASILs are different in each product. Such variation can affect the product-specific fault trees, FMEA, and ASIL allocation generated by HiP-HOPS, and the structure of the product line safety case. Thus, the impact of SPL variation in safety analysis assets is stored and managed in our tool by means of SPL functional, architectural, and component failure models.

Table 1 – HBS-SPL hazard analysis.

Scenario	*Hazard*	*ASIL*
HBS-4WB	No braking four wheels	D
	No braking rear	C
	Value braking	C
HBS-FWB	No braking front	D
	Value braking	D
HBS-RWB	No braking rear	D
	Value braking	D

[3] FMP feature modeling tool: http://sourceforge.net/projects/fmp/

3.2 HBS-SPL Safety Case Architecture

The SPL safety case architecture generated by the tool is illustrated in Fig. 4. The system boundaries specified in the feature model, and the information provided by product line FFM, AFM, and CFM were the inputs required by the tool. The figure presents the structure of the HBS-SPL modular safety case architecture in a GSN modular view organized according to the functional decomposition specified in the feature model of Fig. 3g. Mandatory, optional, and alternative feature/variation-specific argument modules are organized according to GSN variability modeling abstractions described on Fig. 3. These abstractions translate the representation of variation in feature model to variation in the safety case architecture.

The top-level argument module is supported by 'Auxiliary_Battery', 'Powertrain_Battery', 'Mechanical_Pedal', 'Electronic_Pedal', and 'Communication' modules (denoted by 'supported by' relationships between the top-level module and these argument modules). 'Communication' argument module is supported by two argument modules arguing the safety of the 'Bus_1' and 'Bus_2' redundant features. 'Wheel_Braking' is a mandatory argument module supported by 'Brake_Unit1', 'Brake_Unit2', 'Brake_Unit3', and 'Brake_Unit4' alternative wheel braking argument modules encapsulated in the 'WB_Contract' module. The internal structure of the feature/variation-specific argument modules generated by the tool decomposes the argument over the safety of each feature into references to arguments modules (i.e. using GSN Away Goals) arguing the mitigation of the system hazards associated with that feature using the Hazard Avoidance pattern [13]. For example, the 'Brake_Unit1' argument module is decomposed into the 'No_Brake_4W', and 'Val_Braking', alternative hazard mitigation argument modules (Fig. 5a). Each one of these modules is further decomposed into argument modules arguing the mitigation of

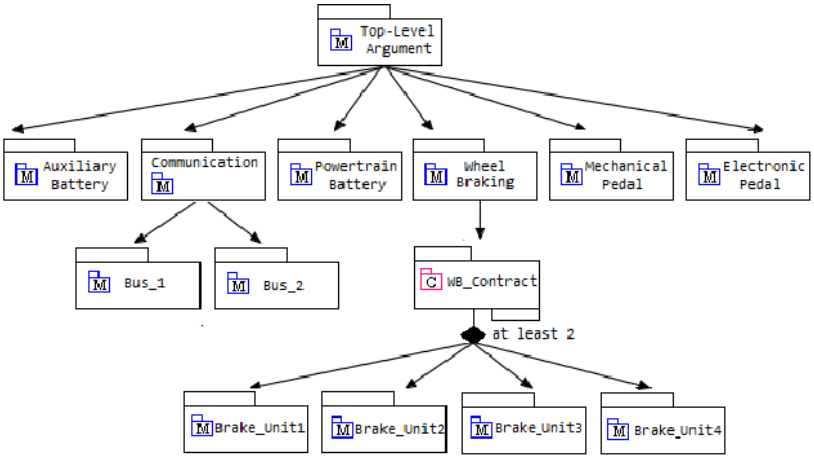

Fig 4. HBS-SPL modular safety case architecture.

component faults contributing to a system hazard. Fig. 4b shows the 'Val_Braking' hazard argument. This argument is decomposed into modules arguing the mitigation of WNC and IWB component failure modes that should be addressed to minimize the hazard effects in a particular product. The selection of these argument modules is

dependent upon the product definition and its operational environment. It may change the values for argument parameters (Fig. 5) changing the structure of the product argument. SPL hazard analyses provide the information required for these parameters.

By analyzing the structure of the argument modules we have identified the importance of the context in the argument structure to encapsulate the variation in product definition (i.e. feature selection) and its operational environment (i.e. context selection) at system (i.e. functional) level argumentation. So, the selection of these elements may change the structure of a product-specific modular safety case architecture. At the architectural level argumentation, context is used to encapsulate the variation in the hazards, their top-level failure conditions, and the allocated safety requirements that are subject to change according to the product definition and context specified at system level argument. Finally, at the component level, the context is used to encapsulate the variation in the component failure modes and safety integrity levels allocated to these failure modes.

The structure of this modular safety case architecture has increased the reusability of safety arguments arguing the safety of variable SPL features. The safety case architecture reflects the decomposition of the system grouping feature-specific safety arguments into cohesive and low-coupled modules. Variable argument modules such as '*Brake_Unit1'* and '*Brake_Unit4'* are encapsulated in a contract module. Thus, the impact of the variation on the selection of these argument modules in a product-specific safety case is isolated from reusable argument structures.

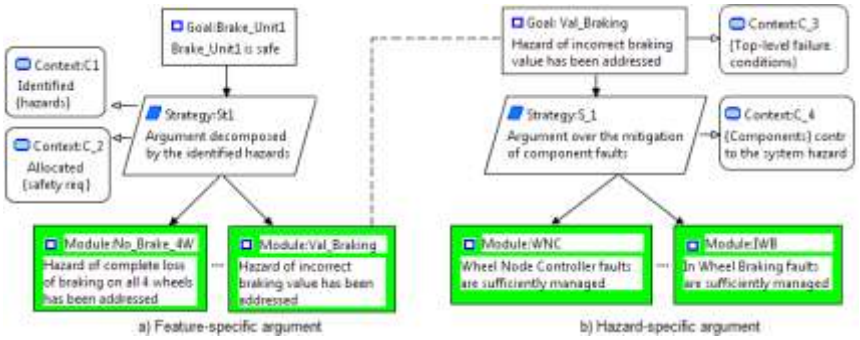

Fig. 5. Brake_Unit1 and hazard arguments.

4 Conclusion

In this paper we proposed a method describing the design and implementation of a tool that supports the automated generation of a modular product line safety case architecture from outputs provided by model-based development, safety analysis, and variability management tools. The SPL safety case architecture generated in this approach is organized into modules reflecting the feature and component boundaries. This architecture has the potential to increase the reuse of safety arguments across SPL products, which is the main novelty of our approach compared with current research on safety cases. The feasibility of the approach was tested in a small case study, but we are aware that more work is needed to assess the value and limitations

of this work and the acceptability of the resultant modular safety cases. It might be the case that only partial argument structures can be constructed using this approach and such structures may need significant intervention by experts to create convincing safety arguments. Further work focuses on evaluating the quality of safety case architectures generated by the proposed method against traditional software metrics like coupling and cohesion.

Acknowledgements

Our thanks to CNPq, grant number: 152693/2011-4, and CAPES Brazilian research agencies for the financial support.

References

1. ACEdit, 2014. Assurance case editor. Available on line: https://code.google.com/p/acedit/
2. Bloomfield, R., Bishop, P. 2010. Safety and assurance cases: Past, present and possible future - an Adelard perspective. In *Proc. of the 18th Safety-Critical Systems Symp.*, Springer-London.
3. Clements, P., Northrop, L. 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.
4. Czarnecki, K., Helsen, S., Eisenecker, U. 2004. Staged configuration using feature models. In 3rd *Software Product-Line Conference*, vol. 3154, Springer-Verlag, pp. 266-283.
5. De Castro, R., Araújo, R. E., Freitas, D. 2011. Hybrid ABS with electric motor and friction brakes. *22nd International Symposium on Dynamics of Vehicles on Roads and Tracks*, Manchester, UK.
6. EUROCAE. 2010. ARP4754A - Guidelines for Development of Civil Aircraft and Systems.
7. Fenn, J., Hawkins, R., Williams, P., Kelly, T., Banner M. G., Oakshott, Y. 2007. The who, where, how, why and when of modular and incremental certification. IET System Safety Conference.
8. Habli, I., Kelly, T., Hopkins, I. 2007. Challenges of establishing a software product line for an aerospace engine monitoring System. In *Proc. of 11th Int'l SPL Conference*, IEEE, pp.193-202.
9. Habli, I. 2009. *Model-Based Assurance of Safety-Critical Product Lines.* Ph.D thesis, Department of Computer Science, The University of York, York, United Kingdom.
10. Habli, I, Kelly, T. 2010. A safety case approach to assuring configurable architectures of safety-critical product lines. *1st Int. Conf. on Architecting Critical Systems*, Springer-Verlag, pp. 142-160.
11. ISO. 2011. ISO 26262: Road Vehicles Functional Safety.
12. Kelly, T. 2007. Using software architecture techniques to support the modular certification of safety-critical systems. *11th Australian workshop on Safety critical systems and software, 69:53-65.*
13. Kelly, T., McDermid, J. 1997. Safety case construction and reuse using patterns. In *Proc. of 16th Int. Conf. on Computer Safety, Reliability and Security*, Springer-London, pp. 55–69.
14. Liu, J., Dehlinger, J., Lutz, R. 2007. Safety analysis of software product lines using stated modeling. *Journal of Systems and Software*, vol. 80, n. 11, pp. 1879-1892.
15. Matsuno. 2014. A design and implementation of an assurance case language. In *Proc. IEEE/IFIP Dependable Systems and Networks (DSN).*
16. Origin Consulting York. 2011. GSN community standard version 1. Available on-line: http://www.goalstructuring notation.info/documents/GSN_Standard.pdf
17. Papadopoulos, Y., Walker, M., Parker, D., Rüde, E., Hamann, 2011. Engineering failure analysis and design optimization with HIP-HOPS. *Journal of Engineering Failure Analysis*, 18.2, 590-608.
18. Steiner, E. M., Masiero, P. C. 2013. Managing SPL variabilities in UAV Simulink models with Pure::variants and Hephaestus. *CLEI Electronic Journal*, v. 16, n. 1.
19. Weaver, R. A. 2003. The Safety of Software – Constructing and Assuring Arguments. PhD thesis, Department of Computer Science, University of York.
20. Weiland, J. 2006. Configuring variant-rich automotive software architecture models. In *Proc. of 2nd IEEE Conf. on Automotive Electronics*, pp. 73-80.