Automatic Allocation of Safety Requirements to Components of a Software Product Line

André L. de Oliveira[*], Yiannis Papadopoulos[**], Luís S. Azevedo[**], David Parker[**], Rosana T. V. Braga*, Paulo C. Masiero*, Ibrahim Habli[***], Tim Kelly[***]

**Mathematics and Computer Science Institute, University of São Paulo, São Carlos-SP, Brazil*
***Department of Computer Science, University of Hull, Hull, United Kingdom*
****Department of Computer Science, University of York, Deramore Lane, York, United Kingdom*
{andre_luiz, rtvb, masiero}@icmc.usp.br, {y.i.papadopoulos, d.j.parker}@hull.ac.uk, l.p.azevedo@2012.hull.ac.uk,
{ibrahim.habli, tim.kelly}@york.ac.uk

Abstract: Safety critical systems developed as part of a product line must still comply with safety standards. Standards use the concept of Safety Integrity Levels (SILs) to drive the assignment of system safety requirements to components of a system under design. However, for a Software Product Line (SPL), the safety requirements that need to be allocated to a component may vary in different products. Variation in design can indeed change the possible hazards incurred in each product, their causes, and can alter the safety requirements placed on individual components in different SPL products. Establishing common SILs for components of a large scale SPL by considering all possible usage scenarios, is desirable for economies of scale, but it also poses challenges to the safety engineering process. In this paper, we propose a method for automatic allocation of SILs to components of a product line. The approach is applied to a Hybrid Braking System SPL design.

Keywords: safety-critical product lines; safety requirements; SILs; requirements allocation.

1. INTRODUCTION

The term Software Product Line (SPL) refers to a software development approach that enables software reuse by allowing the creation of software applications through the composition of common and variable features that address the requirements of a particular domain (Clements and Northrop, 2001). Features represent desired functionality from the user point of view (Lee et al. 2002). Product line design maximizes software reuse across products but must still yield safe individual products and this poses research challenges.

In general safety assessment processes in many industries move into a direction where safety is addressed from the early stages. Safety requirements for a system are captured early on and are progressively allocated to subsystems and components of the architecture. The process must guarantee that, at the end, if the component requirements are met, the system requirements are also met. This type of allocation of requirements is seen as important because it provides a way in which safety is controlled from early stages and is not left to emerge or not at the end. Can we transfer this type of top-down thinking about safety in SPL design? This is the question that is addressed in this paper.

Safety standards use the concept of Safety Integrity Levels (SILs) to assign safety requirements of different stringencies to components of a system. They take the form of Automotive Safety Integrity Levels (ASILs) in the ISO 26262 (ISO, 2011) standard for passenger cars, and Development Assurance Levels (DALs) in aerospace safety standards (EUROCAE, 2010). SILs are assigned early in the system design process at system level, just after the system hazards have been identified. These hazards are given more or less stringent SILs depending on how high are the risks they pose. As the system architecture is being refined, SILs assigned to system-level hazards are iteratively allocated to subsystems and components. ASIL Decomposition is a process that allows for a safety-critical architecture to meet a particular target ASIL assigned to a hazard without all components contributing to the hazard having to meet that target. For example, if a hazard can be caused only when two independent components fail together, these components can share the responsibility of meeting the ASIL allocated to that hazard, rather than each one having to meet the original ASIL.

ISO 26262 defines an integer algebra for ASIL decomposition which is loosely derived from rules about combining probabilities. Each ASIL is equivalent to an integer value: QM (Quality Management) = 0, A = 1, B = 2, C= 3, and D = 4. The ASIL algebra defines that if n components must fail simultaneously to cause a given hazard, the total ASIL assigned to these n components must add up to the ASIL of the hazard they originate. So, two redundant components assuring a function of ASIL D might individually only be required to meet ASIL B because together they produce the total required ASIL value (2 + 2 = 4). Higher ASILs mean higher costs, because meeting more stringent safety requirements typically requires more safety measures, more effort, and higher-quality components.

Therefore, component ASILs could significantly affect both development and production costs. ASIL decomposition allows to efficiently allocate requirements so that we can meet the safety requirements without being unnecessarily stringent or expensive.

More specifically, recent Model-Based Safety Assessment (MSBA) techniques can potentially provide frameworks for SILs allocation, by allowing us to automatically identify combinations of component failures that lead to system hazards, and therefore by locating opportunities for SIL decomposition. HiP-HOPS (Hierarchically Performed Hazard Origin & Propagation Studies) (Papadopoulos et al. 2011) is an advanced MBSA technique that already provides such an approach for ISO 26262. HiP-HOPS implements a combination of model-based, automated Fault Tree Analysis (FTA) process and a Tabu Search (TS) (Azevedo et al. 2013) meta-heuristic optimization algorithm, allowing optimal ASIL allocation and has shown to scale up to complex systems.

Although application of the MBSA process to SPL design would be beneficial, this is not straightforward. As safety is context-dependent, hazards, their causes, and the requirements allocated to SPL components may change according to the selection of SPL variants in a particular product (Habli, 2011). Such variation may change the safety requirements (i.e., the SILs) placed on components in different products of the SPL. Thereby, establishing safety requirements for SPL components requires finding the SILs allocated to those components in different products. If a component is allocated different SILs in different products then the highest requirement must be met for the component to be used safely across SPL products. This type of allocation would allow developers to meet their responsibilities in order to assure the safety of the SPL architecture, and to comply with safety standards, without incurring the unnecessary high costs of complete reanalysis and reallocation of safety requirements as traditionally demanded for each product. However, the establishment of product line component ASILs in a large scale SPL, like a family of automotive powertrain controllers with potentially hundreds of members, can be challenging.

Safety standards do not show how this can be done in large scale, e.g. via automation, and no framework has yet been developed to support the automatic allocation of SILs compatible with, and useful in the context of, SPLs. Indeed, although emerging SIL allocation tools and techniques (Azevedo et al. 2014; Parker et al. 2013; Papadopoulos et al. 2010; Mader et al. 2012; Bieber et al. 2011; Zhang et al. 2010; Lee et al. 2009; Sallak et al. 2008; Dhouibi et al. 2014) provide the capability of automatically allocate SILs to single SPL products, they do not address product lines.

The novelty of this paper is precisely a concept for allocation of SILs to components in product line design, and a method and tool to provide the automated support to apply that concept. The tool is tailored upon HiP-HOPS (Azevedo et al. 2014) and is applied to the automotive domain. The paper is organized as follows. Section 2 provides an overview of HiP-HOPS and its Tabu Search ASIL allocation approach. Section 3 describes the method and tool to automatically allocate SILs to SPL components from the analysis of multi-product SILs allocations. Section 4 presents the case study and evaluation, and section 5 shows related work. Finally, section 6 presents the conclusion and future research.

2. HiP-HOPS AND TABU SEARCH

HiP-HOPS (Papadopoulos et al. 2011) is a method and tool for Model-based Safety Analysis, in which system models showing components and material energy, and data transactions among them are augmented with local failure logic. These models are then analyzed to create forms of safety analysis such as fault trees and Failure Modes and Effects Analysis (FMEAs). The HiP-HOPS tool receives the system description as input in an XML schema, but the various instantiations of the tool, e.g. its connection to MATLAB/Simulink, also provide a failure editor that can be used together with graphical interfaces to augment the model with safety information. Using this editor it is possible to specify hazards related to system malfunctions, and the failure logic of components described mainly as sets of output deviations and how they are caused by logical combinations of internal component failures and deviations of the component inputs. Once the system models have been annotated with hazards and local failure logic, HiP-HOPS synthesizes fault trees for each hazard, and then combines them to create an FMEA for the system.

HiP-HOPS rationalizes allocation and decomposition of ASILs to system components, by showing how combinations of component failures lead to system hazards. For allocation to happen, the potential fault propagation of the system must be defined, to determine which component(s) potentially contribute to each function failure. The rationale is that a component that contributes to a system failure only in conjunction with other components may receive a lower SIL than a component which directly causes the system failure. Thus the design intention of components, and specifically their ability to detect, mask or propagate failures, influences SIL allocation across the architecture; for example, some components may be designed to fail silent in response to failure, possibly transforming a severe failure mode into a less severe failure mode. The contribution of components to hazards can be established using the analysis capabilities of HiP-HOPS and the synthesized fault trees produced by the tool. A fault tree for instance provides the minimal cut sets (i.e. combinations of basic events) that result in system-level hazards, and therefore can be used to identify ASIL Decomposition opportunities.

It is often the case that the failure of the same component is present in multiple cut sets and all of them must be taken into account when finding the most advantageous allocation solutions. Furthermore, with the increase in the number of components within a system, the number of ASIL allocation possibilities increases exponentially. Early implementations of exhaustive algorithms (Papadopoulos et al. 2010) for allocation in HiP-HOPS were found inadequate in coping with the complexity of realistic models. Investigations were, therefore, directed towards meta-heuristics such as genetic algorithms. These optimization algorithms are known to find good solutions, through guided search of a small fraction of the entire search space. In addition, they are known to be robust in dealing with a variety of problems.

Meta-heuristics do not guarantee finding optimal solutions; however, they are capable of providing near optimal allocations within acceptable time spans. Many meta-heuristics exist, and recent research has been testing some of the most popular ones on the ASIL Allocation optimization problem. Tabu Search (Azevedo et al. 2014) has shown promising results in both the quality of the solutions found as well as processing times. The TS extension of HiP-HOPS draws from the work of Hansen and Lih (1996) for reliability optimization and it goes by the name of Steepest Descent Mildest Ascent (SDMA). The method consists of iteratively finding the ASIL that by being decremented reduces the cost of a solution (the steepest descent direction).

3. AUTOMATIC ALLOCATION OF SAFETY INTEGRITY LEVELS TO PRODUCT LINE COMPONENTS

In this work we extended the above capabilities to enable allocation of safety requirements to components of a product line. The concept extends the capabilities of the HiP-HOPS method and tool for application in SPL design. The key idea is the ability to automatically instantiate a large set of products of an SPL from a variable SPL model augmented with possible hazards and the local failure logic of components. Products are then sent in a loop to HiP-HOPS, which performs ASIL allocation for each product. Components may receive different allocations in different products, so the ability to use safely a component across a range of products means selecting the highest allocation given by this analysis.

The process was implemented in a prototype tool developed using Java as a compliment to HiP-HOPS. The tool requires a pre-processing step to support the augmentation of SPL variability models with hazards and failure logic and the enumeration of all products of an SPL via resolution of variability (i.e. product derivation). The latter can be a selective manual process or an automated process. The automated implementation of this feature has been done in MATLAB/Simulink using a range of mechanisms to specify variability and with the support of the product line variability management tool Hephaestus/Simulink (Steiner et al. 2013). Alternative tools include the Common Variability Language (CVL) implemented in Eclipse (Haugen et al. 2008).

Simulink variability patterns (Steiner et al. 2013; Botterweck et al. 2010) representing optional (Enabler subsystems), alternative (Switch blocks), and inclusive-or (Integration blocks) features have been used for modelling variation in the SPL architecture. The variability management tool Hephaestus/Simulink was used in this approach to support the variability modelling in Simulink models using these patterns. The automated derivation of the enumerated SPL product models augmented with hazards and failure logic requires the specification of the configuration knowledge. This provides a set of rules showing how SPL design assets, hazards, and failure logic can be composed in a product. Rules are specified according to feature model constraints. The feature model captures structural or conceptual relationships between common and variable functions of products of a domain (Lee et al. 2002).

SPL configuration knowledge is specified by means of Hephaestus/Simulink by applying the following steps: 1) specify the feature expressions in the scope of the usage scenarios described in the feature model. A feature expression may include a single feature or a combination between two or more features; 2) for each feature expression, determine the SPL 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, the mapping between product line features, design elements, hazards and the allocated safety requirements, and component failure logic is obtained. Additional details on how to use product line variability management tools to specify the configuration knowledge can be found in (Steiner et al. 2013). Finally, the variability management tool was adapted by implementing an instantiation script with the support of a feature model reasoner, in this case the T-wise covering arrays algorithm (Johansen et al. 2012), to automatically derive a set of SPL products according to the constraints specified in the feature model.

Once a range of SPL products have been enumerated from a variable SPL model, the products (i.e., system models annotated with failure information) are then provided to HiP-HOPS for analysis. The ASIL allocations generated by applying HiP-HOPS to the enumeration of the SPL products are the inputs to the tool developed in this paper. HiP-HOPS exports the ASIL allocation of each individual product in an XML file. An extension tool parses each one of these files and performs the analysis. Firstly, the tool analyses the XML files one by one to obtain the ASILs allocated to SPL components in each product. This is done by analysing the ASILs allocated to the failure modes associated to each SPL component in each individual product. The most stringent ASIL allocated to a failure mode associated to a particular component is the component ASIL in that product. For each product, this is repeated for all components belonging to the product. After obtaining the ASILs allocated to product line components in each product, for each SPL component, the analysis is performed as follows: the ASILs allocated to a particular SPL component in different products are analysed in order to verify the most stringent ASIL allocated to that component across the SPL. Thus, we have obtained the ASIL that each SPL component should meet, and the results, i.e. ASILs allocated to all SPL components are exported in an XML file.

4. EVALUATION

The method and tool was applied to a Hybrid Braking System (De Castro et al. 2011) automotive product line (HBS-SPL). Hazard analysis and the definition of local failure logic were performed based on failure logic analysis technique supported by HiP-HOPS, and taking into account the interactions between HBS-SPL components expressed in the feature model. Three HBS-SPL products (i.e., usage scenarios) were considered in the evaluation of the proposed method: HBS four wheels braking (HBS-4WB), HBS front wheels braking (HBS-FWB), and HBS rear wheels braking (HBS-RWB).

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. 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. IWMs have, however, braking torque availability limitations at high wheel speeds or when the powertrain battery is close to full state of charge. EMBs are introduced to 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 HBS-SPL feature model presented in Fig. 1. The feature model was designed using the cardinality-based notation (Czarnecki et al. 2004).

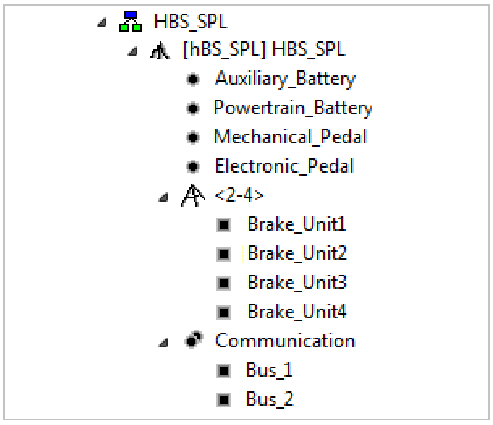
HBS_SPL
[hBS_SPL] HBS_SPL
Auxiliary_Battery
Powertrain_Battery
Mechanical_Pedal
Electronic_Pedal
<2-4>
Brake_Unit1
Brake_Unit2
Brake_Unit3
Brake_Unit4
Communication
Bus_1
Bus_2

Fig. 1 HBS-SPL feature model.

The HBS-SPL feature model 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. The three HBS products described earlier have a common principle: a Mechanical Pedal is responsible for capturing the driver's braking demands; an Electronic Pedal senses these actions and transforms them into braking requests for each wheel that is equipped with braking features; subsequently, it sends these requests via a duplex bus communication system to the Brake Units. Each Brake Unit integrates a Wheel Node Controller that calculates the amount of braking torque to be produced by each actuator. Commands are generated accordingly and sent to the power converters to control the 2 braking devices. While braking, power flows from the IWMs to the Powertrain Battery and from the vehicle's low voltage Auxiliary Battery to the EMBs. The elements of the vehicle's power architecture should be regarded as subsystems that include multiple components – the Powertrain Battery, for example, integrates a Battery Management System (BMS) which is composed by complex hardware and software elements.

Hazards can arise in this system from the interaction between design elements in a range of usage scenarios. Safety requirements placed to a particular HBS-SPL 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 (Lee et al. 2002). In this paper we have limited ourselves to performing the HBS-SPL hazard analysis based only on the SPL feature model. Product line features stand for system functions implemented by design elements (e.g. system, subsystems, components).

Performing a hazard analysis covering all possible scenarios for HBS-SPL would yield voluminous results. Nevertheless, scoping the hazard analysis to a set of products has shown some degree of reuse for safety analysis assets (e.g. fault trees, FMEA, ASIL allocation). 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, as mentioned earlier, the following usage scenarios were established: HBS-4WB; HBS-FWB; and HBS-RWB. These scenarios were analysed from the safety perspective. Table 1 presents the identified hazards, their causes, and the allocated ASILs (Automotive Safety Integrity Levels). Table 1 also presents the association between the hazards and the usage scenarios by means of the column "Scenario".

Table 1. HBS-SPL hazards and ASIL allocation.

Scenario	Hazard	Causes	ASIL
HBS-4WB	No braking four wheels	Omission of all brake unit actuators outputs.	D
	No braking three wheels	Omission of brake unit1, and brake unit2, and brake unit3 actuators outputs.	D
	No braking front	Omission of brake unit1 and brake unit2 actuators outputs.	D
	No braking rear	Omission of brake unit3 and brake_unit4 actuators outputs.	C
	No braking diagonal	Omission of brake unit1 and brake unit4 actuators outputs or Omission of brake unit2 and brake unit4 actuators outputs.	C
	Value braking	Incorrect Value of all brake unit actuators outputs	D
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

It is considered that no braking is being produced in a wheel whenever both braking devices of that wheel (an IWM and an EMB) are omitting their outputs. Braking with an incorrect value happens when at least one of the braking actuators is providing braking torque that is higher or lower than the values demanded. In order to simplify the case study, we have only discussed the allocation of ASILs to product line hazards on the basis of the severity (rather than the full ISO 26262 risk assessment process). In a product line hazard analysis, different ASILs can be assigned to the same hazard considering different usage scenarios for product line components. For example, the ASIL allocated to the "No braking rear" hazard is more stringent in HBS-RWB scenario and less stringent in HBS-4WB. Causes for a particular hazard can also change according to how product line components 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 information are stored by HiP-HOPS in the failure model. From analysis of HBS-SPL hazards (Table 1), 77 failure logic expressions were added to 30 HBS-SPL components; through different fault propagations, the causes described in these expressions combine and give rise to hazards in different product configurations. This process was automated as product models augmented with hazards and local failure logic were sent to HiP-HOPS which created fault trees, failure cut sets, and FMEA results for each HBS-SPL product.

The fault trees generated for the HBS-SPL products provided the input for the HiP-HOPS allocation algorithm. The allocation was performed for each HBS-SPL product based on the following example cost heuristic that expresses the relative cost jumps of developing a component according to the different ASILs: 0 (ASIL QM), 10 (ASIL A), 20 (ASIL B), 40 (ASIL C), and 50 (ASIL D). This expression was used for illustrative purposes, but any other that the system designer finds more suitable can be used instead. We have set the algorithm stopping criteria to 5000 iterations without improvements. All algorithm executions were carried in a computer equipped with an Intel i5 processor clocked at 2.5GHz and 6GB of RAM.

The HBS-4WB, HBS-FWB, and HBS-RWB ASIL allocations provided by HiP-HOPS analysis were the inputs for performing the analysis to allocate ASILs to components. The analysis was also carried in the same computer. The ASILs allocated to 30 HBS-SPL components in three different products were analyzed and the process took 14 seconds to complete. Table 2 presents the ASILs allocated to HBS-SPL components in each product, and the final ASILs allocated to HBS-SPL components (column "ASIL"). Due to space limitations, Table 2 presents ASILs allocated to 16 HBS-SPL components.

ASILs allocated to a particular HBS-SPL component may change according to the product. For example, the ASILs allocated to Brake_Unit1, Brake_Unit1.EMB, and Brake_Unit1.EMB_Power_Converter components are respectively "A", "A", and "A" in HBS-4WB, and "QM", "B", and "B" in HBS-FWB. The ASIL costs related to each HBS-SPL product ASIL allocation was also generated by the tool. The tool also generated the ASIL cost for the HBS-SPL (cell "Cost for the MAX ASIL" on Table 2). The HBS-SPL ASIL cost is higher than the product costs as it represents the worst case where any component of the SPL is designed to be safely used across all products of the SPL.

Table 2. HBS-SPL products HIP-HOPS Tabu Search ASIL decomposition results.

HBS-SPL Component Name	**HBS-4WB ASIL**	**HBS-FWB ASIL**	**HBS-RWB ASIL**	**MAX ASIL**
Auxiliary_Battery	D (4)	D (4)	D (4)	D (4)
Brake_Unit1	A (1)	QM (0)	-	A (1)
Brake_Unit1.EMB	A (1)	B (2)	-	B (2)
Brake_Unit1.EMB_ Power_Converter	A (1)	B (2)	-	B (2)
Brake_Unit1.IWM	A (1)	B (2)	-	B (2)
Brake_Unit1.IWM_ Power_Converter	A (1)	B (2)	-	B (2)
…	…	…	…	…
Brake_Unit4	A (1)	-	B (2)	B (2)
Brake_Unit4.EMB	A (1)	-	B (2)	B (2)
Brake_Unit4.EMB_ Power_Converter	D (4)	-	B (2)	D (4)
Brake_Unit4.IWM	QM (0)	-	B (2)	B (2)
Brake_Unit4.IWM_ Power_Converter	B (2)	-	B (2)	B (2)
Communication_Bus1	B (2)	B (2)	B (2)	B (2)
Communication_Bus2	B (2)	B (2)	B (2)	B (2)
Electronic_Pedal	D (4)	D (4)	D (4)	D (4)
Mechanical_Pedal	D (4)	D (4)	D (4)	D (4)
Electronic_Pedal	D (4)	D (4)	D (4)	D (4)
Cost	**520**	**460**	**470**	**730**

Analysis of these results about the implications on safety requirements of possible usage of components provides useful feedback to the SPL development process, contributing to meeting safety requirements without incurring unnecessary costs.

The tool developed for this work was tested against performance requirements in this case study. The processing time to analyze 72 hybrid braking system SPL products was reasonable, about 4 minute and 40 seconds, considering that the complexity of the analysis has increased substantially as the numbers of the products increased.

5. RELATED WORK

In earlier work, Papadopoulos et al. (2010) proposed an approach to automatically allocate ASILs to subsystems and components of a hierarchical system model according to ISO 26262. The ASIL allocation and decomposition algorithm was implemented in HiP-HOPS (Papadopoulos et al. 2011). The HiP-HOPS ASIL allocation algorithm was further improved with optimization heuristics to reach an optimal allocation. Penalty-based (Parker et al. 2013) and Tabu Search (Azevedo et al. 2014) algorithms were implemented to improve the performance of ASIL allocation in large scale systems.

Mader et al. (2012) proposed an approach for ASIL allocation focused on finding optimal allocations; a linear programming optimization problem is formulated to discover a solution that minimizes the sum of ASILs assigned across the system architecture. Zhang et al. (2010) proposed a workflow for embedded system development, which includes fault trees, FMEA, and ASIL allocation based on a qualitative risk graph method. Dhouibi et al. (2014) introduced a method for ASIL allocation which is based on interpreting the allocation problem as a system of linear equations. Bieber et al. (2011) presented a theory to formalize the ARP 4754a DAL allocation rules (EUROCAE, 2010) and the

DALculator tool to support automatic DAL allocation. Lee et al. (2009) presented an approach based on fault trees and their top-events (i.e., probabilities for failure on demand) to derive SILs for system functions according to IEC 61508. A fuzzy probabilistic SILs allocation technique in compliance with IEC 61508 was also proposed in (Sallak et al. 2008).

Existing tools and techniques for automatic allocation of SILs were not designed to address product lines. These techniques we hope can benefit from the concepts sketched in this paper.

6. CONCLUSION

We described a method for allocation of SILs to components in product line design. A prototype tool was developed which performs automatic ASIL allocation for product line components taking into account their possible usage across the product line. We have discussed, both in theory and through an example, how the use of such a method and tool can potentially reduce the cost of SPL development by allocating less stringent ASILs to SPL components whilst meeting safety requirements. Through this technique, it is possible to specify safety requirements for components anticipating their possible use in a number of products. This work addresses an important issue and extends and automates principles enshrined in modern safety standards to SPL design.

Further work needs to be done to elucidate and explain the preparation and automatic resolution of variable models augmented with failure analyses that can be used in the frame of this method. Additional research is also ongoing to validate this approach in different industrial context.

REFERENCES

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

Azevedo L. S., Parker D., Walker M., Papadopoulos Y., Esteves, A. R. (2013). Automatic Decomposition of Safety Integrity Levels: Optimization by Tabu Search. In Proc. of 2nd Workshop on Critical Automotive applications : Robustness & Safety, of the 32nd SAFECOMP, Toulouse, France.

Bieber, P., Delmas, R., Seguin, C. (2011). DALculus Theory and Tool for Development Assurance Level Allocation. In Computer Safety, Reliability, and Security. Springer Berlin Heidelberg, 43-56.

Botterweck, G., Polzer, A., Kowalewski, S. (2009). Using higher-order transformations to derive variability mechanism for embedded systems. In 2nd Int. Workshop on Model-Based Architecting of Embedded Systems, MODELS, Denver, USA, pp. 68-82.

Clements, P., Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley.

Czarnecki, K., Helsen, S., Eisenecker, U. (2004). Staged configuration using feature models. In 3rd Software Product-Line Conf., v. 3154, Boston, USA, Springer-Verlag, pp. 266-283.

De Castro, R., Araújo, R. E., Freitas, D., (2011). Hybrid ABS with Electric motor and friction Brakes. In 22nd International Symposium on Dynamics of Vehicles on Roads and Tracks, Manchester, UK.

Dhouibi, M. S., Perquis, J. M., Saintis, L. Barreau, M. (2014). Automatic Decomposition and Allocation of Safety Integrity Level Using System of Linear Equations. In Proc. of the 4th Int. Conf. on Performance, Safety and Robustness in Complex Systems and Applications, Nice, France.

EUROCAE. (2010). ARP4754A/ED-79A - Guidelines for Development of Civil Aircraft and Systems.

Glover, F. (1989). Tabu search-part I. *ORSA Journal of Computing,* vol. 1, no. 3, pp. 190-206.

Glover, F. (1990). Tabu Search-part II. *ORSA Journal of Computing*, vol 2, pp. 4-32, 1990.

Habli, I. (2009). Model-Based Assurance of Safety-Critical Product Lines. Ph.D thesis, Department of Computer Science, The University of York, York, UK.

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. Software Product Line Conference, pp.193-202.

Hansen, P., Lih, K. W. (1996). Heuristic reliability optimization by tabu search. *Annals of Operations Research*, vol. 63, pp. 321-336.

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

ISO. (2011). ISO 26262: Road Vehicles Functional Safety.

Johansen, M. F., Haugen, O., Fleurey, F. (2012). An algorithm for generating t-wise covering arrays from large feature models. In Proc. of the 16th Int. Software Product Line Conf., vol. 1. ACM, NY, USA, 46-55.

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

Lee, Y., Kim, J., Kim, J., Moon, I. (2009). A Verification of Fault Tree for Safety Integrity Level Evaluation. In Proceedings of the ICROS-SICE International Joint Conference, pp 5548-5551.

Lin, M. H., Tsai, J. F., Yu, C. S. (2012). A Review of Deterministic Optimization Methods in Engineering and Management, Mathematical Problems in Engineering.

Mader, R., Armengaud, E., Leitner, A., Steger, C. (2012). Automatic and optimal allocation of safety integrity levels. In Reliability and Maintainability Symposium (RAMS), Proceedings-Annual. IEEE, 1-6.

Papadopoulos, Y., Walker, M., Parker, D., Rüde, E., Hamann, R., Uhlig, A., Grätz, U., Lien, R. (2011). Engineering failure analysis and design optimization with HIP-HOPS. *Journal of Engineering Failure Analysis* 18.2, 590-608.

Papadopoulos, Y., Walker, M., Reiser, M. O., Weber, M., Chen, D., Torngren, M., Servat, D., Abele, A., Stappert, F., Lonn, H., Berntsson, L., Johansson, R., Tagliabo, F., Torchiato, S., Sandberg, A. (2010). Automatic allocation of safety integrity levels. In 1st workshop on critical automotive applications: robustness and safety, ACM.

Parker, D., Walker, M., Azevedo, L., Papadopoulos, Y., Araujo, R. (2013). Automatic Decomposition and Allocation of Safety Integrity Levels Using a Penalty-Based Genetic Algorithm. *Recent Trends in Applied Artificial Intelligence*, Springer-Berlin, 449-459.

Sallak, M., Simon, C., Aubry, J. F. (2008). A Fuzzy Probabilistic Approach for Determining Safety Integrity Level. *IEEE Transactions on Fuzzy Systems*, vol. 16, pp 239-248.

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.

Zhang, H., Li, W., Qin, J. (2010). Model-based Functional Safety Analysis Method for Automotive Embedded System Applications. In Proc. Int. Conf. on Intelligent Control and Information Processing, pp 761-765.