Variability management in safety-critical software product line engineering

A. L. de Oliveira[1], R. T. V. Braga[2], P. C. Masiero[2], Y. Papadopoulos[3], I. Habli[4], T. Kelly[4]

[1] Federal University of Juiz de Fora, Juiz de Fora, Brazil
[2]University of São Paulo, São Carlos, Brazil
[3]University of Hull, Hull, United Kingdom
[4]University of York, York, United Kingdom
andre.oliveira@ice.ufjf.br,{rtvb,masiero}@icmc.usp.br,
y.i.papadopoulos@hull.ac.uk,{ibrahim.habli,tim.kelly}@york.ac.uk

Abstract. Safety-critical systems developed upon SPLE approach have to address safety standards, which establish guidance for analyzing and demonstrating dependability properties of the system at different levels of abstraction. However, the adoption of an SPLE approach for developing safety-critical systems demands the integration of safety engineering into SPLE processes. Thus, variability management in both system design and dependability analysis should be considered through SPLE life-cycle. Variation in design and context may impact on dependability properties during Hazard Analysis and Risk Assessment (HARA), allocation of functional and non-functional safety requirements, and component fault analysis. This paper presents DEPendable-SPLE, a model-based approach that extends traditional SPLE methods, to support variability modeling/management in dependability analysis. The approach is illustrated in a case study from the aerospace domain. As a result, the approach enabled efficient management of the impact of design and context variations on HARA and component fault modeling.

Keywords: Variability management, safety-critical systems, dependability.

1 Introduction

Safety-critical systems (SCS) are computer systems in which the occurrence of failures may lead to catastrophic consequences. Due to the benefits of large-scale reuse, Software Product Line Engineering (SPLE) and component-based approaches have been largely adopted by the industry in the development of SCS, especially in automotive [7, 32] and aerospace domains [8, 14]. However, safety-critical systems developed upon an SPLE approach have to address guidance defined in safety standards, e.g., ISO 26262 [18] for automotive, DO-178C [31] and SAE ARP 4754A [10] for aerospace. These standards establish that dependability properties of a SCS should be analyzed and demonstrated at different levels of abstraction before its release for operation. The adoption of an SPLE approach for developing safety-critical systems demands the integration of safety engineering into SPLE processes [14, 28]. Compositional dependability analysis techniques [6, 27, 30] provide the automated support for safety engineering, and seamless integration between system design and dependability analysis. Thus, system design and dependability analysis can be performed in a single model, contributing to reduce the complexity of the product line dependability analysis. Whereas safety-critical SPLE involves safety engineering, variability management in dependability analysis should be considered through software product line (SPL) life-cycle. Dependability analysis can be defined as the identification, early on the design, of potential threats to system reliabil-

ity, availability, integrity, and safety, their potential causes, and measures to avoid or minimize their effects.

In safety-critical SPLE, dependability analysis should be performed aware of the impact of variation in the design and usage context on dependability properties to enable the systematic reuse of both architectural design and dependability information. Variation in the system design and usage context may raise different hazards, with different causes, and different risk that they may pose for the overall safety. As dependability properties may change according to the selection of product variants and their context, variability in dependability modeling should also be managed in safety-critical SPLE. Thus, mapping links between context and design variations and their realization in the dependability analysis, i.e., Hazard Analysis and Risk Assessment (HARA), and component fault modeling, should be defined in the SPL variability model. Existing variability management techniques [1, 15, 17, 32, 33, 34] provide support for managing variation at requirements, architecture, components, source code, and test cases. However, these techniques were not originally designed/or used to support variability management in dependability models developed with the support of compositional analysis techniques, e.g., OSATE AADL [6], HiP-HOPS [30], and CHESS [27].

Variability management in SPL dependability analysis/modeling enables the traceability of the variation in the design and context throughout dependability assets, and the systematic reuse of both architectural components and dependability information. Such reuse contributes to reduce the complexity, effort and costs in performing dependability analysis for specific product variants, since such analysis is not performed from scratch. Therefore, with the support of compositional safety analysis techniques, artefacts such as fault trees and Failure Modes and Effects Analysis (FMEA), required by safety standards for certification of safety-critical systems, can be automatically generated for a given product variant from the reused dependability information. Dependability analysis can only be treated as a product line asset if the variability model contains information about variation points and their realization in the dependability model. Therefore, the dependability model should be included in the SPL core assets, to enable the systematic reuse of dependability information together with other SPL assets, reducing the costs of generating safety assets for a larger number of product variants built around the SPL core assets. Although the reuse of dependability information provided by an SPLE approach is an attractive idea, existing approaches to support it are focused on the reuse of fault trees and FMEA [5, 12, 13, 26, 32], which can be automatically generated from reusable product line dependability analysis information. This paper presents DEPendable-SPLE (DEPendable Software Product Line Engineering), a model-based approach that extends traditional SPLE methods, to address dependability/safety analysis in SPLE, enhancing domain engineering with HARA, functional safety and integrity requirements allocation, component fault modeling activities, the support for variability modeling/management in dependability analysis, and application engineering with variant-specific dependability analysis, product derivation, fault trees and FMEA synthesis. This is required for certification of safety-critical systems in compliance with safety standards, e.g., ISO 26262 Parts 3 and 4 for automotive, and SAE ARP 4754A safety process for avionics. The application of the proposed approach is illustrated in a realistic unmanned aircraft system SPL. The proposed approach distinguishes from traditional SPLE methods by considering the impact of design/usage context variations on dependability analysis, thus, integrating variant management into compositional dependability analysis. Such integration allows the systematic reuse of both SPL architecture and dependability information. The remaining of this paper is organized as follows. Section 2 presents an analysis of existing variability types in safety-critical SPLE

and their relationships. Section 3 presents the DEPendable-SPLE approach to support variability management in dependability analysis, and its application in a realistic aerospace SPL. Sections 4 and 5 detail related work, conclusion, and future work.

2 Product and Usage Context Variation and their Impact

Variability in safety-critical SPLs can be: system (product) or contextual variability [17]. System variability can occur in: capability, operating environment, domain technology, implementation technique, or quality attribute features [19, 24]. Capability comprises features regarding end-user visible characteristics. Operating environment comprises features associated with the target environment where a given software product is operated. Domain technology relates to features representing specific domain techniques or tools that can be used to implement the SPL assets. Implementation techniques are features regarding specific implementation strategies, e.g., redundant or non-redundant control system architectures in the avionics domain. Quality attribute refers to features that the SPL products must address such as usability, maintainability, and data integrity checking. Usage context features relate to where and how the SPL is used. Such classification has been considered through this paper. Combinations among these features act as key driver during safety-critical SPL design, dependability analysis/modeling, and product derivation/instantiation. Such multi-perspective features and combinations among them have a direct influence in SPL architectural decisions and dependability analysis aimed at safety certification. The following subsections presents the Tiriba Flight Control SPL, used through this paper, and an analysis of the impact of variation in product and usage context features on the SPL design and dependability analysis. Such analysis, which is one of the contributions of this paper, was performed by analyzing the following system architectures and their respective dependability models: aircraft braking system [11], door controller [25], Tiriba UAV [33], and automotive braking system [29] product lines. Their architecture models were specified in AADL and Simulink, and their dependability models were specified in AADL/Error Annex, and HiP-HOPS analysis techniques.

2.1 Tiriba Flight Control Product Line

Tiriba Flight Control avionics product line (TFC-SPL) is part of the Tiriba UAV-SPL [2], which comprises a control subsystem with the following goals: to start the flight mode (direct, stabilized, or autonomous), processing and setup flight commands, keeping flight conditions, and executing commands sent by the navigation subsystem. Two different TFC product variants in different usage contexts were considered through this paper. Although Tiriba was originally designed in MATLAB/Simulink, to illustrate the integration of variability management and compositional dependability analysis techniques, TFC architectural and dependability models were specified in AADL and AADL Error Annex [6]. Base Variability Resolution (BVR) toolset [34] and the developed AADL/Error Annex adapter were used to support variability management in both architecture and dependability models. TFC-SPL was designed following an extractive strategy by analyzing the original Tiriba flight control subsystem Simulink model [33]. TFC-SPL comprises the *Pilot Mode* variation point, which contains four pilot mode options/variants: *Manual Pilot* is mandatory, *Assisted Pilot*, *Autonomous Pilot*, and *Autopilot* are optional. *Manual Pilot* mode is a human operator sending commands to the unmanned aircraft vehicle (UAV) from a ground control station. *Autopilot* executes a predefined route. *Assisted Mode* allows the operator sending commands to the UAV configured with *Autopilot*. *Autonomous Pilot* allows the UAV performing actions according to its current

environmental conditions captured by pressure sensors. *Assisted Pilot*, *Autonomous Pilot*, and *Autopilot* can be combined into several different ways, allowing seven different flight control variants. TFC product variants can operate in a range of different contexts defined by combining *Airspace*, *Application*, and *UAV Size* variation points [2]. The composition of *pilot mode* and *usage context* variants leads to 84 different TFC variants. Since it would be prohibitive considering all these variants to perform dependability analysis, only TFC manual and autonomous (TFC-MAT) and all pilot modes (TFC-ALL) product variants, *Controlled and Uncontrolled airspace* context variants, illustrated in Fig. 1a, were considered through this paper.

TFC-SPL architecture comprises 4 subsystems and 14 components, which are composed by 252 model elements and subcomponents. An excerpt of the TFC SysML main block diagram is shown in Fig. 1b. *Mode Switch* encapsulates the *pilot mode* variation point, whilst *Command Switch* encapsulates the variation point inherent to the source from the pilot commands, e.g., manual pilot subsystem, sent to the UAV. *Basic Command Processor (BCP)* and its ports represent the realization of the *Autonomous* pilot mode. *PWM Decoder* output port connected to *Command Switch* block represents the realization of the *Manual Pilot* mode in the design. The realization of *Autopilot* is given by: *Autopilot* subsystem, *PWMDecoder.FlightControls* output connected to *FlightStabilizer* and *CommandSwitch* model elements, and *FlightStabilizer.AutopilotSettings* output port connected to *AssistedModeSwitch.* Finally, *ModeSwitcher.ControlMode* output port connected to *AssistedModeSwitch* and *CommandSwitch* elements represent the realization of the *Assisted Pilot* mode.

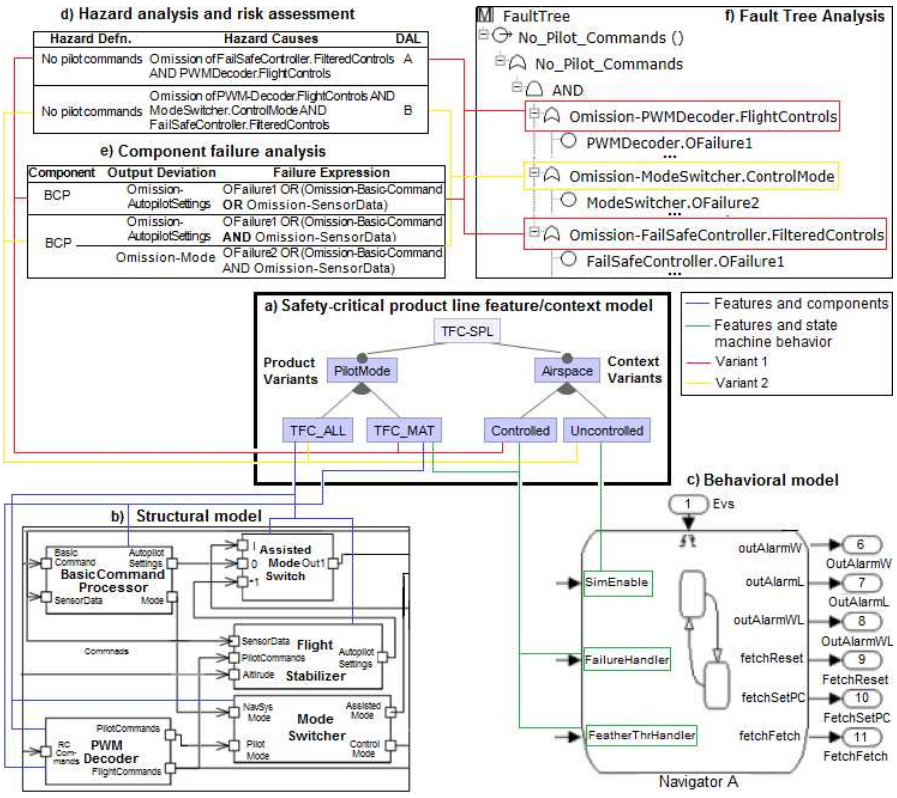

Fig. 1. Impact of product/usage context variations on design and dependability analysis.

2.2 The Impact of Variation on SPL Design

Interactions among usage context features define constraints that may impact on design decisions aiming safety certification, in which different design choices must be taken according to the targeted context. In both non-critical and safety-critical SPLs, variation points and their variants defined in a feature model have a directly impact on the derivation of variant-specific structural/architectural and behavioral models. In the development of safety-critical systems, the system architecture is often expressed in data-flow oriented models, and the system behavior is expressed in Finite State Machines (FSMs). In data-flow oriented architectural models, structural variability can be found in systems, subsystems, components, their ports and connections via flows from input to output ports, which may change according to the targeted usage context. For example, considering the TFC-SPL [33], developed upon Simulink model blocks, the selection of TFC-MAT variant from *Pilot Mode* variation point (Fig. 1a) implies in the selection of *PWM Decoder* and *Basic Command Processor* components, their ports and connections (Fig. 1b) during the product derivation process.

The realization of TFC-MAT and other product variants in the structural model are highlighted in Fig. 1 with blue lines linking features (Fig. 1a) to their realization in architectural components and their ports (Fig. 1b). It is important to highlight that variation in the usage context combined with variation in product features, and isolated variation in the usage context may also impact on the derivation of variant-specific structural and behavioral models. For example, considering the TFC-SPL, the selection of *Pilot Mode* product and *Airspace* usage context variants impacts on the derivation of redundant or non-redundant components in a variant-specific architecture model. Still in TFC-SPL, product features associated with *assisted, autopilot*, *autonomous*, and *manual* pilot modes, i.e., *all pilot modes* (TFC-ALL) product variant (Fig. 1a), are materialized in the architecture by all Tiriba flight control components, their ports and connections (Fig. 1b). Architectural variation inherent to TFC-MAT and TFC-ALL variants and usage context variation can be further propagated throughout Tiriba mission controller behavioral model expressed in a FSM (Fig. 1c).

In safety-critical SPLE, variation in product and usage context features may also impact on the system behavior, which can be expressed in a FSM. Thus, FSM *states*, *state transitions*, and *events* that trigger state transitions may vary according to the selection of the targeted product/usage context. Variation in FSM can directly impact on elements of the system architecture, changing data port values, configuration of components, and their connections. FSM variability can be firstly found in the number and in the structure of the *state flows* associated with different product variants and their usage context. Variation in a *state flow* can be found in its *input* and *output* data, *states*, and *state transitions*. A *State* can have different local *variables* with different values, and it can be involved in different *transitions* according to the targeted product and usage context variants. *State Transition* variation can be found in *source* and *target* states, in the *event* that triggers the transition, its *execution priority* order and *outgoing events*. A transition *Event* may be triggered by different *mode/states* with different *guard conditions*. A *guard condition* is a condition that should be satisfied for the transition from a *source state* to a *target state*. *Transition events* may also have timing constraints. For example, an event is dispatched on an interval of one second after a system failure. *Effects* of a *transition event* represent changes in *state variables*, which might vary according to the targeted product variant and context. Finally, *state transitions* may dispatch different *outgoing events* affecting both behavior and structure of a given product variant. Examples of *outgoing events* are changes on *states* and *variables* from other FSMs, and changes in structural elements such as *systems*, *subsystems*, *subcomponents,* their c*onnections*, and *port* values.

Variation in FSMs can be found in Tiriba UAV optional mission-related features [33]. Variability in the selection of these features was specified in a FSM that defines the Tiriba mission controller using a mechanism of transitions conditioned to variables that define variability, as illustrated in Fig. 1c. Thus, when *Entry Segment Simulation* is selected, a simulation is performed whenever the UAV starts a new mission segment (Fig. 1c), in order to find out the suitable approach to switch between two mission segments. When *Feather Threshold* feature is selected, the UAV route is adjusted whenever the aircraft deviates more than a certain limit from the planned mission route, i.e., correction start state transition in the FSM [33]. Finally, when *Failure Handler* is selected, the UAV is able to return to specific positions where a picture was not captured during the mission [33]. The activation/deactivation of mission controller features are defined by alternative flows with *Boolean* conditional variables (Fig. 1c), which allow enabling/disabling flows and FSM states according to the selected product/context variants. Thus, for each mission-related feature, there is a variable whose value defines which FSM states and transitions will take place in the final product. *SimEnable* is a *Boolean* variable that controls the activation/deactivation of the behavior related to the *Entry Segment Simulation*. When this feature selected, this variable is set with *true* value. The same is valid for selection/deselection of *Failure Handler* and *Feather Threshold* features.

Usage context features may have influence in the selection of product features that impact on the FSMs, changing the system behavior. Thus, when *uncontrolled* airspace context and TFC-ALL variants are chosen, *SimEnable* variable is set *true* activating the transition to the *Simulating* state in a variant-specific mission controller FSM. When *controlled* airspace context and TFC-MAT variants are selected, *Feather Threshold* and *Failure Handler* variables are set *true*, activating the *correction start* and *good simulation* state transitions.

2.3 The Impact of Variation on SPL Dependability Analysis

In addition to the impact of product and usage context variants on architectural and behavioral models in conventional SPLs, such variation may be propagated throughout dependability analysis in safety-critical SPLs. Thus, variation in SPL architectural and behavioral models, defined in product and usage context variants, can be further propagated throughout the SPL safety lifecycle, impacting on HARA, and allocation of safety requirements (i.e., Functional-Safety Concept in ISO 26262). Fig. 1 illustrates the impact of variation in Tiriba SPL design, i.e., architectural (i.e., block diagram in Fig. 1b) and behavioral (i.e., FSM in Fig. 1c) models and usage context (Fig. 1a), on product line HARA and allocation of safety requirements (Fig. 1d), and their propagation throughout component failure modeling (Fig. 1e), Fault Tree Analysis (FTA) and FMEA (Fig. 1f) dependability-related activities required to achieve safety certification. The production of dependability-related artefacts contributes to increase the costs of the system development. Thus, understanding how SPL product and usage context variants impact on dependability analysis contributes to achieve the systematic reuse of both design and dependability artefacts, reducing the certification costs of individual SPL products. Combinations among product and usage context variants may be useful to derive scenarios, which can be used to guide dependability analysis in safety-critical SPL architectures. Thus, different failure conditions can lead to system-level failures (*hazards*) with different probability, severity, criticality levels, and different safety requirements can be allocated to avoid/minimizing hazard effects on the overall safety according to design choices and targeted contexts. Safety requirement is the required risk reduction measures associated with a given system failure or component failure. Variability types in dependability analysis are detailed in the following.

Variability in HARA. During hazard analysis, different *hazards* and *hazard causes* can emerge according to the targeted product and usage context variants. Fig. 1d shows an example of variation in Tiriba SPL hazard analysis and risk assessment. The causes for *no pilot commands* hazard are omission failures in *FailSafeController* and *PWMDecoder* component outputs when TFC-MAT product and *controlled* airspace usage context variants are chosen (Fig. 1a). On the other hand, omission failures in *FailSafeController*, *ModeSwitcher*, and *PWMDecoder* component outputs are the causes for the occurrence of the same hazard when TFC-ALL variant and *uncontrolled* airspace context are selected. In risk assessment, variation can be found in *probabilistic criteria*, e.g., *likelihood* and *severity* in SAE ARP 4754A avionics standard, used to classify the risk posed by each system hazard for the overall safety. In Tiriba SPL risk assessment, different risk probability and severity were assigned to *no pilot commands* hazard, according to the targeted product and usage context variants. Thus, the probability of occurrence of an omission of pilot commands, i.e., *no pilot commands* hazard, is 10^{-9} per hour of operation, with a *catastrophic* severity when TFC-MAT product and *controlled airspace* usage context variants are chosen. On the other hand, the probability of occurrence of this hazard is 10^{-7} per hour of operation with a hazardous *severity* when TFC-ALL product and *uncontrolled airspace* context variants are chosen.

Variability in the Allocation of Safety Requirements and Integrity Levels. Still in HARA, after classifying the risk posed by each identified hazard and contributing component failure mode, variation can be found in the allocation of *functional safety requirements* and *Safety Integrity Levels* (SILs) to mitigate the effects of system or component failures on the overall safety. Variation in *functional safety requirements* can be found in architectural decisions that must be taken to eliminate or minimizing the effects of a system or component failure, which might change according to the targeted product and usage context variants. For example, in the Tiriba UAV product line, the control system architecture can be *Redundant* when *controlled airspace* usage context is chosen. A *non-redundant* architecture can be adopted when the UAV is intended to operate in an *uncontrolled airspace* [2].

Variation in the allocation of SILs relates to the variation on the mitigation mechanisms to handle the risk posed by a given system hazard, component failure mode, or component, which might change according to the targeted product and usage context variants. This may impact on the SPL development process, in which different system engineering activities, e.g., verification, validation, and testing should be carried out to address the targeted level [3]. Process-oriented safety standards, e.g., DO-178C and ISO 26262, define a set of safety objectives that should be addressed per SIL. Safety objectives define a set of activities to be performed and artefacts to be produced. DO-178C defines five levels of integrity named Development Assurance Levels (DALs). Level A is the highest stringent integrity, and level E is the less stringent. ISO 26262 automotive standard also defines five levels of safety integrity where QM is the less stringent and level D is the stringent. Addressing higher stringent SILs demand the most stringent safety objectives, system engineering activities, and software artefacts, increasing the development costs. Allocating less stringent SILs to less-critical SPL components and more stringent SILs only to highly critical components can contribute to reduce the SPL development costs. An example of variation in integrity levels is observed in the allocation of DALs to mitigate the effects of the occurrence of *no pilot commands* hazard, during the Tiriba SPL dependability analysis (see Fig. 1d). So, DAL A should be assigned to mitigate the effects of this hazard in TFC-MAT variant operating in a *controlled airspace*. On the other hand, DAL B should be assigned to mitigate the effects of this hazard in TFC-ALL variant operating in an *uncontrolled* airspace. Such variation may be further propagated throughout

the decomposition of SILs allocated to hazards throughout contributing failures modes and components. Variation in SIL decomposition, which can be performed with automated tool support [30], is outside the scope of this paper. It is important to highlight that when developing reusable components, all variability aspects of a component should be considered from the initial stages of the SPL lifecycle, and the most stringent SIL assigned to that component in different contexts should be assigned to that component to ensure its safety usage across the SPL. Thus, product and contextual variability will not change the mitigation mechanisms for that component in specific product variants.

Variability in Component Failure Modeling. In the safety lifecycle, component failure modeling is intended to identify how components contribute to the occurrence of potential system-level failures identified during hazard analysis. Variation in component failure modeling can be found in *component output deviations* that contribute in some way for the occurrence of system failures, which might change from a targeted product/context variant to another. Different *input deviations*, *internal failures*, or combinations among them (named *failure expression)*, which contribute to the occurrence of a given *output deviation*, may also be raised in different product/context variants. Variation in the design, i.e., in architecture and behavior models, may also impact on how component failures *propagate* throughout other components. Thus, *output deviations* of a given component may be propagated throughout different components according to the chosen product/context variants. Different product/context variants lead to different connections among components, via their ports, so it may change the way in which failures are propagated throughout the system architecture.

Variation in HARA can be propagated throughout how components contribute to the occurrence of system hazards. Tiriba *Basic Command Processor* (BCP) component failure modeling, shown in Fig. 1e, illustrates examples of variation in *output deviations* and combinations among *component failures* leading to *output deviations* in two different product/context variants. *Omission-autopilotSettings* output deviation may be raised when TFC-MAT product and *controlled* airspace context variants are chosen, whilst one additional output deviation, i.e., *omission-mode,* can also be raised when TFC-ALL product and *uncontrolled* airspace context variants are chosen. Such variation can also be propagated throughout the causes that lead to the occurrence of a given component output deviation. The causes of an output deviation can be stated in a failure expression using logical operators (AND, OR, NOT) that describe how combinations among internal and input failures of a component may lead to the occurrence of an output deviation. Such variation can be found in the causes of the *Omission-autopilotSettings* output in both product variants. An internal omission failure in BCP component or an omission in both BCP inputs can raise an *Omission-autopilotSettings* output in TFC-ALL variant. Conversely, an internal omission failure in BCP or an omission failure in one of its inputs may raise an *Omission-autopilotSettings* in TFC-MAT variant.

Probabilistic criteria values, e.g., *likelihood*, *severity*, *failure and repair rates*, which can be assigned to a given hardware component, may also vary according to the targeted product and usage context variants. An example of such variation was found in the assignment of failure rates to *Barometric Processor* hardware component from the Tiriba UAV-SPL. The component failure rate is 10^{-9} per hour of operation when TFC-MAT variant is chosen, and 10^{-7} per hour when TFC-ALL variant is selected. Variation in component failure modeling can be further propagated throughout **fault trees and FMEA** dependability artefacts, which can be automatically generated, with the support compositional dependability analysis techniques [6, 27, 30], from HARA and component failure modeling. Such variation can be seen in fault tree gates and nodes (Fig. 1f), and in the way how components contribute to hazards in a FMEA

table from different product variants [5, 28, 29, 32]. Thus, variation in hazard causes can be evidenced in the structure of a fault tree as illustrated in Fig. 1f, in which *PWMDecoder* and *FailSafeController* component output deviations are top-level failures of *no pilot commands* fault tree when TFC-MAT variant is chosen. On the other hand, an output deviation in *ModeSwitcher* component together with other two aforementioned output deviations, are the top-level failures of *no pilot commands* fault tree when TFC-ALL product variant is chosen. Finally, variation in HARA, allocation of safety requirements, component failure modeling, fault trees and FMEA results are propagated throughout the structure of a variant-specific assurance case. An assurance case is a defensible, comprehensive, and justifiable argument supported by a body evidence which demonstrate that the system acceptably safe to operate in a particular context [22]. Assurance case is recommended by standards for certifying safety-critical systems in both automotive [18] and aerospace [10] domains. The analysis of the impact of product and context variants in assurance case is outside the scope of this paper.

3 DEPendable-SPLE

Interactions among product and usage context features can be used to establish safety certification criteria that might impact on SPL architectural decisions and dependability analysis. This section presents DEPendable-SPLE, a model-based approach that extends traditional SPLE methods, with support for HARA, allocation of safety requirements and component failure modeling dependability-related activities, variability modeling/management in both SPL domain and application engineering phases as illustrated in Fig. 2. In domain engineering, four steps related to dependability analysis and variability realization modeling were defined. The last step extends the variability modeling step, established in conventional SPLE methods, by considering feature realization in the dependability model. This is intended to support the systematic reuse of dependability information in SPL application engineering.

In SPL application engineering, the following steps were defined: product requirements analysis (i.e., product feature modeling), product derivation, which extends conventional SPLE product derivation process by linking features to their realization in the dependability model, and dependable-related activities: product dependability analysis, and fault trees and FMEA synthesis required to achieve the systematic reuse of dependability information, reducing product safety certification effort and costs. Compositional dependability analysis [6, 27, 30] and variability management techniques [1, 32, 33, 34] can be used to support DEPendable-SPLE dependability analysis and variability management steps, both in domain and application engineering. DEPendable-SPLE is applicable independently from the underlying variability management and compositional dependability analysis techniques. It should be used together with traditional SPLE methods [4, 19, 23] for developing dependable safety-critical SPL architectures. The steps shown in Fig. 2 are described and illustrated, by considering TFC

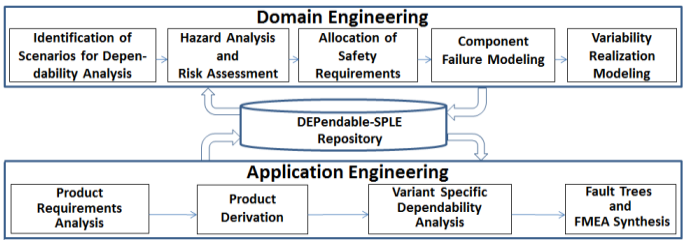

Fig. 2. An overview of DEPendable-SPLE approach.

-MAT and TFC-ALL product variants from TFC-SPL, in the following.

3.1 DEPendable-SPLE: Domain Engineering

In the first step, dependability analysis scenarios are defined from the analysis of interactions among product and usage context features. After scoping SPL dependability analysis to a set of targeted scenarios, HARA, allocation of safety requirements, and component failure modeling dependability-related steps are performed iteratively and incrementally for each scenario. Finally, features are linked to their realization in architecture and dependability models during safety-critical SPL variability realization modeling.

Identification of Candidate Scenarios for SPL Dependability Analysis. *Inputs:* the feature model containing the specification of product and usage context features and their interactions, SPL structural and behavioral models, and analyst's domain knowledge. *Purpose:* identifying, from the analysis of interactions among product and usage context features, variants (*scenarios*) relevant for the stakeholders, by combining features that may be considered to guide SPL dependability analysis. The identification of scenarios encompasses the following tasks: *i)* to identify combinations among product features, which represent system functions and their interactions in structural and behavioral models, to derive product variants; *ii)* for each identified product variant, combinations among features are analyzed to establish potential usage contexts in which the given product variant can operate; and finally, *iii)* by combining the identified product/context variants, different scenarios can be derived. For example, from the analysis of TFC-SPL product/context variants, a range of candidate scenarios can be derived. Scenarios might support safety analysts in extracting the required domain knowledge to perform SPL dependability analysis. However, since it would be prohibitive performing such analysis covering all product/context variants, the analyst's domain knowledge, and relevant product variants for the stakeholders can be used as criteria to assess candidate scenarios to perform dependability analysis [3, 28, 29]. *Outputs:* combinations among product and usage context variants relevant for the stakeholders. In TFC-SPL, the TFC-MAT/*Controlled* and TFC-ALL/*Uncontrolled* scenarios were considered to perform dependability analysis.

Hazard Analysis and Risk Assessment. *Inputs:* targeted scenario relevant for the stakeholders, SPL structural/architectural and behavioral models. *Purpose:* After choosing a given scenario earlier identified at the previous step, HARA can be performed aiming at identifying combinations among component failures leading to system-level failures named hazards. Hazards can be specified by means of logical expressions involving potential failures in SPL architectural components that might lead to system failures. These failures are generally stated in terms of failure types that typically include: *omission*, *commission*, *value*, *early* or *late* failure modes. Firstly, interactions among core architectural components are analyzed to identify potential hazards that can emerge from such interactions. Later, optional, alternative, and mutual inclusive/exclusive architectural components, representing variation defined in the targeted scenario, are analyzed to identify potential hazards that can emerge in such scenario. This step results in a list of variant-specific hazards. Next, risk assessment is performed to estimate/classify, based on probabilistic risk tolerability criteria defined in the targeted safety standard, the risk posed by each identified hazard. *Outputs:* list of identified hazards and risk classification for each analyzed scenario. TFC-SPL HARA was performed with the support of domain experts, by considering TFC-MAT and TFC-ALL and their usage context, which were analyzed from the perspective of SAE ARP 4754A risk assessment process. During HARA, variation in *no pilot commands* and *value pilot commands* hazards and their risks were identified in both two aforementioned variants (Fig. 1d) as described in Section 2.3. *No*

pilot commands can emerge in a TFC variant whenever both pilot modes, e.g., manual pilot and autopilot, are omitting their outputs. An incorrect value for pilot commands can emerge when all pilot modes provide wrong flight commands, e.g., wrong coordinates.

Allocation of Safety Requirements. *Inputs:* HARA results. *Purpose:* From the analysis of HARA results, functional safety requirements and SILs are allocated aimed at eliminating or minimizing hazard effects on the overall safety. SILs should be allocated to each identified hazard according to their risk classification. Moreover, SILs allocated to system hazards can be further decomposed throughout contributing components and their failure modes. Such decomposition can be performed with automated support of SIL decomposition genetic/meta-heuristic algorithms [30]. Finally, allocation of functional safety requirements aims at identifying system functions that can be added to the system architecture for eliminating or minimizing the impact of a hazard or a component failure on the overall safety. Redundancy is an example of functional safety requirement. When a new functional safety requirement is added to the SPL architecture, dependability analysis should be performed again to evaluate the impact of the newer functionality on the overall safety in the context of the targeted scenarios. *Output:* system functions added to the system architecture, and a set of SILs assigned to mitigate hazards and component failures. In TFC-SPL, DALs from different stringencies were assigned to hazards in different TFC variants (Fig. 1d), as described in Section 2.3.

Component Failure Modeling. *Inputs:* SPL architecture model and HARA results. *Purpose:* From the analysis of the identified hazards that can emerge in a particular scenario, assumptions about how architectural components can fail and contribute to the occurrence of each identified hazard can be made. In this step, firstly, a particular component is chosen, followed by the identification of potential output deviations that can contribute to the occurrence of each identified hazard. Next, potential causes of each identified output deviation should be specified by analyzing potential combinations among component internal failures and input deviations that may lead to the occurrence of each output deviation. Such analysis continues whilst there are architectural components to be analyzed. *Output:* a set of component failure data showing how components can contribute to the occurrence of system hazards in each targeted scenario. In TFC-SPL, different component failures may contribute to the occurrence of the identified system hazards in different TFC variants and usage contexts. Thus, component failure modeling was carried out, and 106 failure logic expressions were added to 47 Tiriba flight control model elements. Fig. 1e illustrates an example of variation in the specification of failure logic for the BCP component. Different output deviations may be raised from this component in different product variants. Omission of *AutopilotSettings* and *Mode* outputs may be raised, contributing to the occurrence of hazards when TFC-ALL variant is chosen. However, only an omission of *AutopilotSetting*s output contributes to the occurrence of hazards when TFC-MAT product variant is chosen. Besides, different combinations among component input deviations and internal failures contribute to the occurrence of *Omission-autopilotSettings* output in both TFC-MAT and TFC-ALL variants (column *Failure Expr.* in Fig. 1e). Thus, when variability in the dependability model is solved for TFC-ALL variant, only component failure logic associated with this variant is included in a variant-specific dependability model. Variation in the DALs allocated to TFC hazards is further propagated throughout DALs allocated to mitigate the effects of contributing component failure modes. Variant-specific SIL decomposition can be performed automatically with the support of metaheuristics and genetic algorithms [30]. Additionally, product line SIL decomposition can be derived from the automated analysis of multiple variant-specific SIL decomposition results [28]. Early on the design, it can support the elaboration of a cost-effective safety-critical SPL

development process. Finally, different component failure data lead to different fault propagations, i.e., combinations of component failures leading to hazards. Such variation is then propagated throughout fault trees and FMEA. In order to support the systematic reuse of the TFC-SPL dependability model, the impact of design and usage context variations on dependability information is managed during the variability modeling.

Variability Realization Modeling. *Inputs*: SPL feature, architectural, behavioral, and dependability models. *Purpose*: This step expands the *variability modeling* step defined in traditional SPLE methods [4, 23], aimed at establishing mappings between features to their realization in the architecture, to enable variability management in the dependability model. Thus, in this step, mappings linking product/context features to their realization in architectural, behavioral, and dependability models are defined. It can be performed with the support of extensions from existing variant management techniques. This step encompasses the following tasks: *i)* specification of design and dependable-related variation points and their variants from the analysis of product/context features, and scenarios used to guide dependability analysis; *ii)* mapping design variants to their realization in the SPL architecture by: defining model elements to be included/excluded when design variants are chosen/resolved during product derivation; and finally, *iii)* mapping dependable-related variants to their realization in the dependability model by: linking each scenario considered during dependability analysis to the corresponding HARA, risk assessment, allocation of safety requirements, and failure data associated with each variant-specific component. *Output*: The end of this step yields a variability model, which links features to their realization in both architecture and dependability models. TFC-SPL feature and variability models were specified with the support of BVR toolset and BVR AADL Error Annex adapter. Details about how to configure a BVR variability model for a given SPL can be found elsewhere [34]. BVR fragment substitutions were defined to show how variability in the TFC-SPL AADL architecture and Error Annex models are solved when TFC-MAT/*Controlled* and TFC-ALL/*Uncontrolled* structural and dependable-related variants are chosen. Table 1 illustrates placement and replacement fragments, and fragment substitutions associated with TFC-MAT/*Controlled* airspace dependable-related variant. The acronyms in table columns respectively represent: variation point (*VP*), fragment substitution (*FS*), fragment type (*F*), and component failure data (*CFD*). An excerpt of mappings linking dependable-related variants to their realization in the dependability model, defined in the TFC-SPL variability model, is shown in Table 1. So, when TFC-MAT variant and its context are selected, as defined in a placement fragment, HARA results and component failure data associated with other TFC variants are removed from the dependability model. On the other hand, as specified in the replacement fragment, TFC-MAT related HARA results and component failure data are included in the final dependability model. Later, TFC-MAT related fragment substitution is created by combining both placement/replacement fragments. The final TFC-SPL variability model comprises 8 fragment substitutions, 2 placements, and 8 replace-

Table 1. Pilot model/usage context variant and its realization in the dependability model.

VP	FS	F	HARA Results	CFD
Pilot Mode/ Airspace	TFC-MAT/ CONTROLLED	P	MAS-NoPilotCommands, MAS-ValuePilotCommands, MAP-NoPilotCommands, MAP-ValuePilotCommands, ALL-NoPilotCommands, ALL-ValuePilotCommands	ALL-BCP, MAS-BCP, MAS-PWD
		R	MAT-NoPilotCommands, MAT-ValuePilotCommands	MAT-BCP, MAT-MS, MAT-PWD

ment fragments. Four fragments substitutions are associated with structural variability, and four fragment substitutions are associated with variability in the dependability model.

3.2 DEPendable-SPLE: Application Engineering

Product Requirements Analysis. *Input:* SPL feature model. *Purpose:* In this step, architectural and dependable-related variants, defined in the SPL feature model, which addresses product-specific requirements, are chosen. *Output:* product feature model. **Product Derivation.** *Inputs:* SPL feature and augmented variability models, product feature model, and SPL assets, in this case, architecture, behavioral, and dependability models. *Purpose*: Variability specified in SPL architecture/dependability models are solved with the support of a variability management tool, and a product variant is then derived. Whereas the available variant management techniques do not provide native support for managing variability in the dependability model, in this paper, BVR toolset [34] was adapted to enable support for variability management in AADL Error Annex dependability models. The BVR adapter extends OSATE AADL model editor to enable BVR communicating with OSATE model editors to manage variability in AADL structural/behavioral/dependability models. Since BVR is built upon Eclipse Modeling Framework [9], the adapter was implemented as an Eclipse-based plugin. Due to page limitation, BVR AADL extension is not discussed in this paper. During TFC-SPL product derivation, for each product variant, the following artefacts were input to BVR: TFC-SPL feature model, an instance model, variability model, and SPL AADL/Error Annex structural/dependability models. Outputs: Variant-specific AADL/Error Annex models. Variability management in TFC-SPL enabled the systematic reuse of almost 100% of dependability information, produced during domain engineering, in the derivation of each TFC variant.

Variant-Specific Dependability Analysis. *Inputs:* product feature model, product architectural and behavioral models, analyst's domain knowledge. *Purpose:* After product derivation, variant-specific dependability analysis can be performed by following the steps defined in domain engineering. Such analysis focuses on identifying the impact of variant-specific system functions, added to the reused product architecture model, on the overall system safety. When variant-specific system functions are added to the SPL architecture, its corresponding dependability information should be added to the SPL repository. *Outputs:* Enhanced product architecture and dependability models, and in some cases, feedback to the SPL process, enhancing SPL architecture and dependability models. **Fault Trees and FMEA Synthesis.** *Inputs:* The reused variant-specific architecture model, enhanced with specific dependability information, is the input for automatic synthesis of fault trees and FMEA, supported by compositional analysis techniques [6, 27, 30]. *Purpose:* Generating FTA and FMEA, which are evidence required by safety standards, e.g., ARP 4754A, to achieve safety certification, from the reused dependability information. The accuracy of the generated variant-specific fault trees and FMEA is dependent whether dependability analysis activities were performed aware of the impact of variation in the design and context. *Outputs*: FTA and FMEA are used to demonstrate that the system architecture addresses the safety requirements. Variant-specific TFC AADL structural and dependability models were input to generate fault trees for 8 variant-specific hazards, and FMEA results were synthesized from the fault trees. Fig. 1f shows an excerpt of *no pilot commands* fault tree generated for TFC-ALL variant, which illustrates the impact of SPL variation on hazard causes (see Section 2.3). Such variation is further propagated throughout failure modes that indirectly contribute to the occurrence of this hazard. In FMEA, different component failures might directly/indirectly contribute to the occurrence of

hazards in each TFC variant. Variation in fault trees and FMEA are further propagated throughout assurance cases, which can be generated from these assets with the support of model-based techniques [16]. The application of DEPendable-SPLE steps enabled the systematic analysis of the impact of variation in TFC-SPL product/context features in both SPL design and dependability analysis. Such analysis increased the precision of dependability analysis information produced in domain engineering, and enabled the systematic reuse of dependable-related information early on the SPL safety lifecycle, in comparison with conventional SPLE methods, which emphasize the analysis of the impact of product/context variation only on the SPL design, and reuse of development artefacts.

4 Related Work

Research on variability management in dependability assets is split into extensions of traditional safety analysis techniques, e.g., FTA and FMEA, to suite SPLE processes [5, 12, 26], and model-based techniques [7, 20, 21, 32]. The most notable work in the first category is the extension of software FTA (SFTA) to address the impact of SPL variation on dependability analysis [5, 12]. In such approach, each leaf node of a SFTA is labeled with the commonality or variability associated with that leaf node. This approach is built upon a technique for developing a product line SFTA in domain engineering, and a pruning technique for reusing SFTA in application engineering. Product Line SFTA was later extended to integrate SFTA results with state-based models [26]. Such extension allows mapping SFTA leaf nodes into components, and modeling the behavior of each component in a state chart. PL-SFTA and its extension consider FTA as a reusable asset. However, FTA can be automatically generated from dependability information produced in domain engineering, e.g., HARA and component failure modeling. Thus, variability management in dependability properties earlier on FTA and FMEA synthesis, as presented in this paper, enables the systematic reuse of dependability information, and traceability of dependable-related variation throughout SPL safety life-cycle.

In the second category, Schulze *et al.* [32] have proposed an approach, by integrating commercial *Medini* ISO 26262 safety analysis and *pure::variants* tools, to support variability management in functional safety-related assets, which was evaluated in an automotive case study. Their approach is based on a referencing model, which maps problem-domain features with artefacts in the solution space, in this case, requirements, FTA, and safety goals. Schulze *et al.* approach was further extended with a process for model-based change impact analysis of variability into automotive functional safety [20]. This process combines variability management techniques with safety engineering and software configuration management activities to achieve a complete safety assessment. This process supports change impact analysis in the following scenarios: when a specific variant shows undesired behavior and it needs to be fixed, in cases where an innovative function requires an extension of an existing system function, and the function behavior is changed and it should be analyzed, and when a newer optional function is developed. In the same way as Schulze *et al* approach and its extension, our approach is built upon a variability model, linking problem-domain context and product features with artefacts from the solution space, e.g., components and their failure data. Although Schulze *et al.* [32] approach provides support for variability management in functional safety assets, they didn't emphasize the management of the impact of contextual variation in architecture, HARA and component failure modeling, as presented in this paper. In addition, our approach is applicable to domains other than automotive, and it is independent from the underlying tooling support. Nevertheless, Schulze *et al.* approach [32] and its extension [20] also presented a

good and efficient solution for variability/change management in functional safety. Kaßmeyer *et al.* [21] presented a systematic model-based approach integrated with the change impact analysis approach [20]. Kaßmeyer *et al.* tool-supported approach combines requirements engineering and architectural design, safety analysis, and variant management tools, allowing seamless safety engineering across product variants by representing safety artefacts in a homogeneous UML-compliant model notation. In their approach, HARA and component fault modeling is performed by annotating the UML model in the same way as DEPendable-SPLE approach. As part of Kaßmeyer *et al. approach,* Domis *et al.* [7] have extended Component Integrated Component Fault Trees (C^2FT) technique with variation points and integrated it with UML via a profile into Enterprise Architect tool. Although Kaßmeyer *et al.* [21] approach and its extension [7] also provides a good solution for variant management in functional safety; it is dependent upon specific commercial tools. On the other hand, DEPendable-SPLE can be applied independently of the underlying tooling support.

5 Conclusion

This paper has presented DEPendable-SPLE model-based approach to support variant management in dependability analysis. Such approach enables the systematic reuse of SPL architecture and dependability models in application engineering. DEPendable-SPLE is applicable independently from the underlying variant management and dependability analysis techniques. In this paper, the approach was applied with the support of OSATE AADL & Error Annex architectural modeling and compositional dependability analysis tool, BVR toolset, and OSATE AADL BVR adapter. This adapter was developed by the authors to enable BVR managing variability in AADL architectural/dependability models. These tools were used to support dependability analysis and variant management steps for the TFC-SPL. DEPendable-SPLE supports the analysis of the impact of design and context variations on dependability analysis. Thus, Tiriba product/context variants were linked to their realization in architecture and dependability models. Further, multiple variant-specific architecture/dependability models were automatically generated during product derivation with the support of BVR tool. The systematic reuse of the dependability model is achieved early on SPL safety process. DEPendable-SPLE enabled the systematic reuse of almost 100% of TFC dependability information, produced in domain engineering, in the derivation of each one of the four TFC product variants. It contributed to reduce the effort and costs in performing dependability analysis for a specific product variant. With the support of compositional techniques, in this case, OSATE AADL, FTAs and FMEA were generated from the reused AADL architecture/dependability models. This paper also presented an analysis of the impact of design/context variations on architecture and dependability analysis. Further work on this topic is focused in detailing how variability in AADL architecture/error models is specified and managed. Further work also intends to investigate the impact of design/context variations on SIL allocation and development processes of safety-critical SPLs. Finally, we also intend to investigate the usage of model-driven techniques to generate variant-specific assurance cases, and the potential of SIL decomposition techniques in supporting SPL architectural decisions.

Acknowledgments. CNPq grant number: 152693-2011-4, and CAPES research agencies.

References

1. Big Lever. 2016. Gears. Available on-line: http://www.biglever.com.

2. Braga, R. T. V., Trindade Jr., O., Branco, K. R. L. J. C., Lee, J. Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In: *Proc. of the 16th SPLC*, 1–10, (2012).
3. Braga, R. T. V., Trindade Jr., O., Branco, K. R. L. J. C., Neris, L.O., Lee, J. Adapting a software product line engineering process for certifying safety critical embedded systems. In *Proc. 31st* SAFECOMP, 352-363, (2012).
4. Braga, R. T. V., Branco, K. R. L. J. C, Trindade Jr., O., Masiero, P.C. The ProLiCES approach to develop product lines for safety-critical embedded system and its application to the unmanned aerial vehicles domain. *CLEI Electronic Journal*, v. 15, n.2, 1-12, (2012).
5. Dehlinger, J., Lutz, R. Software fault tree analysis for product lines. In *Proc. of the 8th IEEE HASE*, USA, (2004).
6. Delange, J., Feiler, P. Architecture fault modeling with the AADL error-model annex. In *Proc. of the 40th EUROMICRO*, Verona, 361-368, (2014).
7. Domis, D., Adler, R. Becker, M. Integrating variability and safety analysis models using commercial UML-based tools. In *Proc. of the 19thSPLC*, ACM, USA, 225-234, (2015).
8. Dordowsky, F., Bridges, R., Tschope, H., Implementing a software product line for a complex avionics system. In *Proc. of the 15th Int. SPLC*, IEEE, 241-250, (2011).
9. ECLIPSE. 2016. Eclipse Modeling Framework Project. Available on-line: http://www.eclipse.org/modeling/emf.
10. EUROCAE. 2010. ARP4754A - guidelines for development of civil aircraft and systems, EUROCAE.
11. EUROCAE. Aircraft wheel braking system. Available: https://github.com/osate/examples/tree /master/ARP4761
12. Feng, Q., Lutz, R. Bi-directional safety analysis of product lines. *Journal of Sys. and Sw*,v.78,n.2, 111-127, (2005).
13. Gómez, C., Peter, L., Sutor, A. Variability management of safety and reliability models: an intermediate model towards systematic reuse of component fault trees. In *Proc. 29th SAFECOMP,* Springer, v. 6351, 28-40, (2010).
14. Habli, I., Kelly, T., Hopkins, I. Challenges of establishing a software product line for an aerospace engine monitoring system. In *Proc. of the 11th SPLC*, IEEE, Japan, 193-202, (2007).
15. Haugen, O., Moller-Pedersen, B., Oldevik, J., Olsen, G. K., Svendsen, A. Adding standardized variability to domain specific languages. In *Proc. of the 12th Int. Software Product Line Conf*., IEEE, 139-148, (2008).
16. Hawkins, R., Habli, I., Kolovos, D., Paige, R., Kelly, T., Weaving an assurance case from design: a model-based approach. In Proc. of the 16th HASE, Daytona Beach, IEEE, p.110-117, (2015).
17. Heuer, A., Pohl, K. Structuring variability in the context of embedded systems during software engineering. In: *Proc. of the 8th Workshop on Variability Modelling of Software-intensive Systems*, ACM, (2014).
18. ISO. 2011. ISO 26262: road vehicles functional safety.
19. Kang, K. C., Kim, S., Lee, J., Kim, K., Jounghyun Kim, G., Shin, E. Form: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Eng.,* v. 5, 143–168, (1998).
20. Käßmeyer, M., Schulze, M., Schurius, M. A process to support a systematic change impact analysis of variability and safety in automotive functions. In *Proc. of the* 19th *SPLC*, ACM, NY, USA, 235-244, (2015).
21. Käßmeyer, M., Moncada, D. S. V., Schurius, M. Evaluation of a systematic approach in variant management for safety-critical systems development. In *Proc.13thInt.Conf. Embedded and Ubiquitous Comp*., IEEE, 35-43, (2015).
22. Kelly, T., MCdermid, J., Safety case construction and reuse using patterns. In: *Proc. of the* 16th SAFECOMP, Springer-London, LNCS, p. 55–69, (1997).
23. Krueger, C. Variation management for software production lines. In: *Proc. 2nd SPLC*, v.2379, 37-48,(2002).
24. Lee, K., Kang, K. C. Usage context as key driver for feature selection. In *Proc. 14th SPLC*, v.6287, 32–46p, (2010).
25. Leveson, N. Door control system. Available on-line: https://github.com/osate/examples/tree/master/Train
26. Liu, J., Dehlinger, J., Lutz, R. Safety analysis of software product lines using stated modeling. *Journal of Systems and Software*, v. 80, n. 11, 1879-1892, (2007).
27. Mazzini, S., Favaro, J., Puri, S., Baracchi, L. CHESS: an open source methodology and toolset for the development of critical systems. In Join Proceedings of EduSymp, pp. 59-66 (2016).
28. Oliveira, A. L., Braga, R., Masiero, P. C., Papadopoulos, Y., Habli, I., Kelly, T. Model-based safety analysis of software product lines. *International Journal of Embedded Systems*, Inderscience Publishers, (2016).
29. Oliveira, A. L., Braga, R. T. B., Masiero, P. C., Papadopoulos, Y., Habli, I., Kelly, T. A model-based approach to support the automatic safety analysis of multiple product line products. In *Proc. of the 4th Brazilian Symp. on Computing Systems Eng.,* Brazil, IEEE, 7-12, (2014).
30. Papadopoulos, Y., Walker, M., Parker, D., Rüde, E., Hamann. Engineering failure analysis and design optimization with HIP-HOPS. *Journal of Eng. Failure Analysis,* v.18, i.2, 590-608, (2011).
31. RTCA. DO-178C software considerations in airborne systems and equipment certification, (2012).
32. Schulze, M., Mauersberger, J., Beuche, D. Functional safety and variability: can it be brought together? In *Proc. of the 17th Int. SPLC*, ACM, NY, USA, 236-243, (2013).
33. Steiner, E. M., Masiero, P. C., Bonifácio, R. Managing SPL variabilities in UAV Simulink models with Pure::variants and Hephaestus. *CLEI Electronic Journal*, v. 16, n.1, 1-16, (2013).
34. Vasilevskiy, A., Haugen, Ø., Chauvel, F., Johansen, M. F., Shimbara, D. The BVR tool bundle to support product line engineering. In *Proc. of the 19th Int. Software Product Line Conf.,* ACM, New York, USA, 380-384, (2015).