Variability management in safety-critical systems design and dependability analysis

André Luiz de Oliveira1,2 | Rosana Braga2 | Paulo Masiero2 | David Parker3 | Yiannis Papadopoulos3 | Ibrahim Habli4 | Tim Kelly4

1Department of Computer Science, Federal University of Juiz de Fora, Juiz de Fora, Brazil
2Institute of Mathematics and Computer Science, University of Sao Paulo, Sao Carlos, Brazil
3Faculty of Computer Science, University of Hull, Hull, UK
4Department of Computer Science, University of York, York, UK

Correspondence
André Luiz de Oliveira, Department of Computer Science, Federal University of Juiz de Fora, Juiz de Fora, Brazil.
Email: andre.oliveira@ice.ufjf.br

Funding information
CNPq, Grant/Award Number: 152693-2011-4; CAPES Brazilian research agencies; Post-doctoral program in Computer Science from the University of Sao Paulo, Grant/Award Number: 2016.1.1027.55.4

Abstract

Safety-critical systems are of paramount importance for many application domains, where safety properties are a key driver to engineer critical aspects and avoid system failures. For the benefits of large-scale reuse, software product lines (SPL) have been adopted in critical systems industry. However, the integration of safety analysis in the SPL development process is nontrivial. Also, the different usage contexts of safety-critical systems complicates component fault modeling tasks and the identification of potential hazards. In this light, better methods become necessary to estimate the impact of dependability properties during Hazard Analysis and Risk Assessment. Existing methods incorporating the analysis of safety properties in SPL are limited as they do not include hazard analysis and component fault modeling. In this paper, we present the novel DEPendable Software Product Line Engineering (DEPendable-SPLE) approach, which extends traditional SPL processes to support the reuse of safety assets. We also present a detailed analysis of the impact of product and context features on the SPL design, safety analysis, and safety requirements. We applied DEPendable-SPLE to a realistic case study from the aerospace domain to illustrate how to model and reuse safety properties. DEPendable-SPLE reduced the effort of safety analysis for certifying system variants.

KEYWORDS
dependability analysis, reuse, safety certification, safety-critical systems, software product lines

1 INTRODUCTION

Systems are safety-critical, where the occurrence of failures may have catastrophic consequences, eg, damages to property, environment, injuries, or loss of life. Commonly, the design choices made in different contexts can impact on quality properties, such as reliability, availability, safety, or security. Thus, it is important that dependability analysis be performed to identify potential threats that may lead the system to unsafe states. Safety-critical systems can be found in many complex domains with existing standards/considerations that provide guidance and compliance requirements that must be met for their development, eg, ISO 26262 for automotive, DO-178C and SAE ARP 4754A for the aerospace domain. As part of this, safety guidelines establish that dependability properties of critical systems must be analyzed at different levels of abstraction, eg, from requirements to component levels, before their release for operation.

One of the ways for building such systems is to adopt a Software Product Line (SPL) approach that integrates dependability engineering activities in the development process. Existing compositional techniques provide automated support for dependability engineering processes with seamless integration between system design and dependability analysis. Such integration contributes to a reduction in the complexity of dependability engineering activities and facilitates large-scale reuse, as illustrated in industrial examples from automotive and aerospace domains. However, even when adopting SPL strategies, the reuse of dependability information can still be challenging. Existing approaches only support the reuse of fault trees and Failure Modes and Effects Analysis (FMEA) as key artefacts to identify how system failures are propagated in the design of the system. Fault trees and FMEA artefacts can be automatically generated from the reusable dependability information. Thus, the effort, and costs of performing dependability analysis for specific system configurations/variants can be reduced, since such analysis is not performed from scratch.
One of the main benefits of using dependability information is improving the verification of dependability properties in different usage contexts. This paper presents an approach to facilitate the reuse of dependability information to handle unpredictable hazards and component failures more efficiently and ensure dependability properties. This research extends conventional SPL approaches,18-20 to include dependability information, focusing on the analysis of safety and reliability properties and their impact on traditional software product line life cycle. The analysis of other dependability properties, eg, security and availability, is outside the scope of this research. The major improvements on our previous work22 consist of a detailed description of our approach and Tiriba case study, and a deeper analysis of the impact of design and usage context variations on system architecture and behaviour, allocation, and decomposition of safety requirements,1 component fault modeling and propagation, and FMEA results.

The remainder of this paper is organized as follows. Section 2 presents the background knowledge of the Tiriba unmanned aircraft system, used in the case study, and the research challenges. Section 3 presents our approach consisting of an extension of SPL activities to support dependability analysis for safety-critical systems. We illustrate in Section 4 a case study from the aerospace domain using the Tiriba Flight Control product line in different usage contexts. In Section 5, we present a discussion and an analysis of the impact of system and usage context feature variation on system design and dependability analysis. In Section 6, we describe the related work, and in Section 7, we present the conclusions and future work.

2 | BACKGROUND AND CHALLENGES

This section presents a general overview of the Tiriba unmanned aircraft system25 to provide some context for the case study and the approach of this research. Tiriba is a small low-cost autonomous electrical airplane developed by AGX Company that is used for performing predefined missions and targeted applications. Such applications include agriculture (eg, detection of crop diseases), environment monitoring, traffic monitoring, topographical surveys, inspection of transmission lines, and defense (Figure 1 [left]). The Tiriba unmanned aircraft system, shown in Figure 1 (right), mainly comprises four subsystems: inertial, pressure, control, and navigation units. Each subsystem is allocated to a dedicated microprocessor, in this case, PIC32 80 MHz microcontrollers.

The inertial unit is responsible for estimating, at runtime, the current Unmanned Aircraft Vehicle (UAV) position by receiving information from inertial sensors and Global Navigation Satellite System (GNSS). The pressure subsystem aims at providing the UAV altitude, vertical airspeed, and aero-dynamical speed based on air pressure. The navigation unit is intended to guide the UAV along a route, to follow rules defined by the mission planner and accomplish the tasks associated with each waypoint, eg, taking pictures from a specific location. The Tiriba control unit or flight control, which is used in the case study, provides the following functions: (a) flight control mode, (b) flight and setup commands, and (c) checking flight conditions that are sent by the navigation system. Finally, the ground control station (GCS) sends and receives mission data to/from the UAV.

The Tiriba UAV was originally developed as a single system,23 but it has evolved into an SPL24 to address the subsequent demand for newer, slightly different, versions. The first versions of Tiriba were developed targeting missions over private farms in Brazil. The Tiriba SPL24 comprises 104 features grouped into usage context, quality attributes, capability (or system/product feature), operating environment, domain technology, and implementation technique feature categories defined in Lee and Kang’s modeling approach.26 Dependencies can be specified among these features using mapping tables showing inclusion (v), exclusion (x), and selection (o) relationships between features. For example, the mission abortion feature should be included when Tiriba UAV is flying in a controlled/segregated airspace. Relationships between system and usage context features can also be specified in separated feature models.28-30 This paper focuses on the analysis of dependability properties, in terms of safety and reliability, of the Tiriba Flight Control System SPL (ie, the control unit shown in Figure 1 [right]), detailed in Section 4.

Nowadays, one of the main problems in safety-critical systems is enabling the systematic reuse of the dependability assets and verifying dependability properties at different levels of abstraction (eg, requirements, architectural, and component levels) in a range of different configurations and usage contexts. This is important for identifying potential hazardous behaviours and their causes, mitigating the effects of safety threats, and understanding how hazards can be propagated in the architecture and components of safety-critical systems. As systems like Tiriba can operate in a range of different contexts and under different circumstances, it is important to support dependability analysis in different scenarios. More specifically, we focus on the following challenges: (a) Ch1: enabling the systematic reuse of dependability information and (b) Ch2: ensuring safety and reliability properties of critical systems, eg, pilot modes, in a range of unexpected situations.

3 | DEPENDABLE-SPLE APPROACH

In this section, we describe the DEPendable-SPLE approach as an extension of our previous work.22 Firstly, we describe the impact of dependability in SPLs and outline how conventional software product line processes can be extended to support dependability engineering activities. Secondly, we describe how to model the variability of safety information across different products of the modified SPL life cycle.

---

*Potential source of harm caused by the malfunctioning behavior of a system or a set of systems that implement a function.2
†Freedom from unacceptable risk posed by hazardous failures.21
‡The probability that a component will perform its intended function satisfactorily for a prescribed time and under stipulated environmental conditions.23
§The required risk reduction measure(s) associated with a given system hazard or component failure.23
3.1 Dependability engineering in software product lines

Software Product Line Engineering (SPLE) and component-based approaches provide the benefits of large-scale reuse, contributing to reducing the time-to-market and costs and increasing the quality of components. For these benefits, SPLE approaches have been largely adopted by industry in the development of critical systems, especially in automotive\(^{10,11}\) and aerospace\(^{5,12}\) domains. In the same way as critical systems built upon conventional approaches, critical systems built upon an SPL approach have to consider guidance defined in safety standards/considerations, eg, automotive ISO 26262 and aerospace DO-178C and SAE ARP 4754A. Standards establish that dependability properties of a critical system should be analyzed at different levels of abstraction before its release for operation. At the requirements level, during hazard analysis, the potential threats and the risk that they pose to overall system safety are identified and determined. At the design level, the propagation of system failures throughout architectural subsystems should be analyzed, during component fault modeling, using failure logic analysis and/or Fault Tree Analysis (FTA)\(^{17}\) techniques. Finally, at the component level, it is necessary to identify how components can contribute, directly or indirectly, to the occurrence of system failures (named hazards) using FMEA or Failure Modes, Effects, and Criticality Analysis (FMECA) techniques.

In a safety-critical SPLE process, variation in the system design and usage context¶ may impact on dependability properties. Thus, different hazards, with different causes and risks that they pose to the overall safety, can be raised according to design choices and targeted contexts. Additionally, safety requirements, in the form of system functions and/or Safety Integrity Levels\(^*\) (SILs) with different degrees of stringency should be allocated so the effects of system or component-level failures can be mitigated according to the targeted design and usage context.

Since system dependability properties may change from one targeted product configuration/variant and context to another, dependability analysis for re-configurable systems or systems built upon an SPL approach should be performed with awareness of variation in the design and usage context. This is needed to ensure the safe use of reusable architectural subsystems and components in a range of targeted product configurations and contexts. Thus, we should manage variability in dependability assets from the early stages of a safety-critical SPLE processes. A way to do that is to add mappings to the application variability model that link context and design variants, specified in the domain variability model, to their realization in the dependability assets (eg, hazard analysis and component fault models).\(^{31}\)

Existing variant management techniques\(^{11,32-36}\) provide support for managing variability of requirements, architecture, components, source code, and test case assets. However, these techniques were not originally designed or used for supporting variability management in system models enhanced with dependability information, such as those specified with the support of compositional modeling techniques like OSATE AADL\(^7\), HiP-HOPS\(^8\), or CHESS.\(^9\)

The management of variability in the dependability information enables automated/semi-automated traceability of variation in the design and usage context throughout the dependability assets. This leads to the systematic reuse of both design (architecture and behavior) and dependability information. The systematic reuse of dependability assets may contribute to reducing the complexity, effort, and costs of performing safety analysis for specific system variants, or versions of a particular variant, since such analysis does not need to be performed from scratch. Therefore, we can automatically generate, with the support of compositional techniques,\(^7-9\) fault trees, and FMEA dependability artefacts for a specific system variant from the reusable dependability information. FTAs and FMEA assets are required by cross-domain safety standards, regulations, and considerations, eg, aerospace SAE ARP 4754A, as evidence for certification of critical systems and the safety approval of their components.

It is only possible to consider dependability analysis artefacts as part of the SPL core assets if the application variability model\(^{31}\) contains information about variation points/variants and their realization into the dependability model (these can be in the form of annotations in

---

¶ Usage context relates to where and how a system, a system function, a system configuration is used.\(^{26}\)

* It specifies the reliability, eg, in terms of likelihood and severity, associated with a given system or component-level failure or system function.
the system model). Thus, the dependability model should be included in the SPL core assets to enable the systematic reuse of dependability information together with other SPL assets (eg, requirements and design).

The management of variability in dependability models can contribute to reducing the costs of generating safety assets for a larger number of system variants built upon the SPL core assets. Although existing variant management techniques, eg, Base Variability Resolution (BVR), were not designed to support variability in the dependability model, they can be adapted to enable support for variability management and derivation of product-specific system models enhanced with dependability information.

### 3.2 Adapting conventional SPL approaches to support dependability engineering

Software Product Line Engineering for critical systems is distinguished from conventional SPL approaches as they consider the impact of design choices and usage context on system dependability properties throughout the SPL life cycle (see Figure 2), in compliance with safety standards/considerations. In Figure 2, related conventional SPL and dependability engineering activities are grouped into labeled boxes. Modified SPLE sub-sub-phases are highlighted, and dependability activities added to conventional product line processes are filled with gray. Dotted activities represent issues outside the scope of this paper. In the problem space domain definition, ie, during SPL requirements engineering, dependable/safety-related features and their relationships with system/design features should be identified and specified, together with design and usage context features. This results in the product line feature model, as highlighted in Figure 2 (i).

From the perspective of safety-critical SPLs, interactions among different types of features (eg, dependable-related, system/design, and usage context) can be used to establish safety certification criteria that might impact on SPL architectural/design decisions aiming at achieving a target safety level and dependability properties. After specifying a preliminary SPL design, we perform an analysis of interactions between dependable, system, and context features to derive candidate scenarios (see Figure 2 (ii)) to guide engineers during hazard analysis for identifying potential threats (hazards) and estimating their risk for the overall safety across a set of system variants. In addition to conventional SPLE processes, we should include Hazard Analysis and Risk Assessment (HARA), along with allocation/identification of safety requirements, dependability engineering activities between SPL requirements and initial stages from the design in the domain engineering phase (see Figure 2 (iii)).

In the SPL architecture and implementation, ie, in the solution space, we should specify the realization of dependable-related system features, eg, redundancy, and their relationships with other features in the product line architectural and/or behavioural models and component implementations. Later, dependable-related architecture and component assets should be added to the SPL core assets. Additionally, we should incorporate component fault modeling dependability engineering activity into the SPL architecture subphase (see Figure 2 (iii)).

**FIGURE 2** Conventional software product line processes enhanced with safety/dependability engineering activities
During component fault modeling, engineers identify how components can contribute to the occurrence of hazards in each scenario under analysis, thus, generating a set of varying component fault models. We can perform both hazard analysis and component fault modeling in parallel to the SPL design and component implementation with the support of compositional techniques. It is important to highlight that domain-specific hazard analyses and component fault models are part of full-up safety considerations, ie, safety/dependability assumptions valid for a set of targeted system variants (scenarios).

Still in the domain engineering phase, in order to achieve the systematic reuse of SPL core assets, conventional SPL processes demand the specification of mappings linking system design features to their realization in the SPL architecture along with components in the variability model, during integration of variability information shown in Figure 2 (ii). A variability model or application variability model establishes traceability links between application requirements, specified in the feature model, and domain artefacts such as architectural components and test cases. Moreover, safety-critical SPL processes also demand the specification of mappings in the SPL variability model that link dependable-related features to their realization in the dependability model (ie, HARA information and component fault models). This is necessary to enable support for the systematic reuse of dependability information in the application engineering phase.

Dependable-related features also impact on the development of test cases for testing dependable SPL components. Thus, in the safety-critical SPL, we should add dependability engineering activities to the domain engineering phase. Variability management and other traditional SPLE activities, eg, domain modeling, should be extended for considering the impact of SPL variation on system safety/dependability properties.

In the safety-critical SPL application engineering phase, we need to extend product-specific requirements, ie, product feature modeling in the problem space, by considering the specification and selection of dependable-related features, which impact on both design and dependability assets, together with the specification and selection of system and usage context features. Still in the application engineering phase, we need to modify the variability resolution process of a safety-critical SPL by considering mappings linking dependable-related features to their realization in both SPL design and dependability assets defined in the variability model. This is required to enable the generation/derivation of both design, HARA information, and component fault model safety assets, ie, solution space, for a specific system variant according to the feature selection. The goal is to achieve the systematic reuse of both HARA information and component fault models, reducing potential effort and costs for achieving product certification and component approval in compliance with domain-specific safety standards and/or considerations.

Finally, we should add the same dependability engineering activities from the domain engineering phase, eg, hazard analysis and component fault modeling, together with Fault Tree Analysis and FMEA Synthesis, to the application engineering phase (see Figure 2 (iii), (iv), and (v)). This is necessary to enable support for the analysis of dependability properties of a specific system variant obtained after product derivation (ie, PL Architecture Customization).

### 3.3 DEPendable-SPLE: An overview

Here, we present the DEPendable-SPLE approach, which extends traditional SPLE methods with the provision of support for safety/dependability engineering and variability management in dependability assets (dependability model).

The DEPendable-SPLE approach activities and their associated work products, in both domain and application engineering phases, are illustrated in Figure 2. In this paper, we focus on dependability engineering for safety-critical SPL architectures, so modifications in traditional SPLE activities such as domain and application requirements engineering, design, and testing are not detailed. Instead, we have added four dependability engineering activities to the domain engineering phase, and we have extended traditional SPLE variability management to enable support for the management and resolution of variability in the dependability model. In the application engineering phase, the product derivation process (ie, PL architecture customization) was extended to allow the resolution of variability on the SPL dependability model. This supports the generation of variant-specific design and dependability assets according to chosen features defined in the application feature model.

Although existing variant management techniques were not originally designed to support variability in a system model enhanced with dependability information, we have developed an adapter to the BVR toolset to enable support for managing variability in AADL system models enhanced with AADL error annex dependability annotations. This adapter allows BVR to communicate with OSATE model editors. The detail about the AADL-BVR adapter is outside the scope of this paper.

The DEPendable-SPLE approach is structured to comply with safety assessment processes established in cross-domain standards/considerations. Enhancing SPL processes with safety engineering life cycle activities may contribute to reducing the effort and costs of achieving certification of critical systems along with component approval in compliance with standards/considerations, eg, ISO 26262. Parts 3 and 4 for automotive, and SAE ARP 4754A for safety process for avionics. In our approach, SAE ARP 4754A safety processes or ISO 26262 Parts 3 and 4 activities, ie, PL Architecture box in Figure 2 (ii), are conducted after feature modeling and identification of scenarios for hazard analysis (Domain Modeling and Requirements Engineering box in Figure 2 (ii)). It is important to highlight that there is a two-way process between the standards and domain analysis: The choice of the standards constrains the domain model and vice versa. State-of-the-art model-based and compositional dependability analysis techniques can be used to support engineers performing dependability activities through the SPL life cycle. The DEPendable-SPLE approach is applicable independently from the underlying variant management and compositional safety/dependability analysis techniques. Each of the activities is described in the following sections.

---

1. It means compliance to regulations in avionics, since there is no safety certification in aviation.
3.4 DEPendable-SPLE: Domain engineering phase

In the first step of the domain engineering phase, dependability analysis scenarios are defined from the analysis of interactions among product/system and usage context features specified in the SPL domain model. After scoping SPL dependability analysis to a set of targeted scenarios, dependability-related steps such as hazard analysis, along with the allocation of safety requirements, and component fault modeling, are iteratively and incrementally performed for each scenario. Finally, features are linked to their realization in architectural and behavioral models, plus their dependability artefacts as part of safety-critical SPL variability modeling. Each domain engineering approach step is detailed in terms of its inputs, purpose, and outputs.

3.4.1 Domain modeling and requirements engineering

In this subphase, we describe the elicitation of safety/dependability requirements, how we can model the variability of these requirements and other system features, and how we perform the identification of scenarios for hazard analysis. The elicitation of safety/dependability requirements is not the main goal of our approach. The purpose of this subphase is to analyze different system configurations belonging to the target safety-critical system's domain, eg, avionics, to identify common requirements. However, it is not possible to guarantee that these common requirements belong to safety aspects of the system before performing hazard analysis. After this activity, we perform the feature modeling task.

Feature modeling

In feature modeling, we specify system and usage context commonality and variability in both SPL feature and context models. System variability can happen in capability, operating environment, domain technology, implementation techniques, or quality attribute features. 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 technologies or tools that can be used to implement the SPL core assets. Implementation techniques are features regarding specific implementation strategies, eg, redundant or nonredundant control system architecture features in the avionics domain. Quality attributes refer 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 software product line is used.

Features can have relationships with each other. These relationships can be of the following types: requires (R), excludes (E), works for (WF), or work against (WA), which can be classified as n-to-m. Thus, a particular feature can require, exclude, work for, or against one or more other features. There are even cases where two or more features may require, exclude, or are working for/against a single feature. We have considered this feature classification for this paper. Combinations among system and usage context features act as a key driver during software product line design for safety-critical systems, with dependability analysis and modeling and product derivation (instantiation). Such multiperspective features and combinations have a direct influence on SPL design decisions and dependability analysis when aimed at achieving approval and compliance with safety standards/considerations.

The SPL system feature model comprises features visible to the end user, eg, capability, operating environment, domain technology, implementation technique, and quality attribute features. The context model comprises features describing variation on the SPL environment, ie, where and how system features are intended to be used. The strategy of separating SPL domain modeling into two interconnected feature models is based on the Hartman and Trew feature modeling approach and its refinements. Such approaches allow a clear specification of interactions, eg, requires and excludes, among system and usage context features that may impact on both design and dependability assets. The outputs of this activity are the SPL feature and usage context models and their connections. For example, we can have an SPL system/product feature model (FM) with F1 mandatory and F2 optional system features, and a context model (CM) with CX1 and CX2 mutually exclusive usage context features, and a relation implies that the choice of the CX2 context feature triggers the selection of the F2 system feature.

Usage context features in isolation or when combined with system features can impact on the product line design and dependability analysis in the domain engineering phase. Thus, variation in the selection of system and usage context features may lead to different design choices, resulting in different system and component-level failures during SPL dependability analysis. We can use combinations among system and usage context features to propagate/specify variation in safety information in the feature model. A detailed description of an approach to support the specification of safety-related features and their interactions on feature modeling is outside the scope of this paper.

Identification of scenarios for product line hazard analysis

The last step of the domain modeling subphase is the identification of scenarios for hazard analysis. The inputs for this activity are the SPL feature and context models, containing the specification of system and usage context features and their interactions; a preliminary SPL design, produced during earlier stages of SPL architecture subphase (see Figure 2 (iii)); and analysts' domain knowledge. The purpose of this activity is identifying, from the analysis of interactions among system/product and usage context features, a set of system variants (scenarios) relevant for the stakeholders by combining system and usage context features. These scenarios can be further considered to guide engineers during SPL dependability analysis.

The identification of scenarios encompasses the following tasks: (a) identifying combinations among system/product features, which represent system functions, and their relationships with elements from architectural and behavioral models, to derive system/product variants; (b) for each identified system variant, we analyze combinations among features to establish potential usage contexts in which the given system variant
can operate; and finally, (c) we derive different scenarios by combining the identified system and usage context variants. However, since it would be prohibitive to perform such analyses covering all system/usage context variants, we can use the analysts’ domain knowledge and relevant product variants for the stakeholders as criteria to assess candidate scenarios to be considered when performing SPL dependability analysis.62437

The outputs of this activity are combinations of system and usage context variants, ie, scenarios, relevant for the stakeholders. Thus, considering the SPL feature model (FM) with F1 mandatory and F2 optional system features, and a context model (CM) with CX1 and CX2 mutually exclusive features, and a relation CX2 includes F2, we have the following scenarios: SC1 = F1, CX1, and SC2 = F1, CX2, F2. We can further consider these scenarios during SPL hazard analysis.

3.4.2 | Product line architecture

In the safety-critical SPL development process, system and usage context features and their interactions define constraints that may impact on design decisions aimed at achieving safety certification. Thus, different design choices must be taken according to the selection of system and usage context features. In both non-critical and safety-critical software product lines, variation points and their variants defined in the SPL feature and context models have a direct impact on the derivation of product-specific architectural and behavioral models. In the development of critical systems, the system architecture is often expressed in data-flow oriented models, and the system behavior is often expressed in Finite State Machines (FSMs). So, in this subphase, variability in the SPL domain model propagates throughout the design.

Considering FM and CX product line feature and context models, and SC1 and SC2 scenarios, we can have a hypothetical aerospace SPL architectural model (PLA) comprising S1 and S2 mandatory subsystems, and a C1 optional component, where S1 and S2 subsystems represent the realization of the F1 system feature and CX1 context feature (SC1 scenario). On the other hand, C1 component represents the realization of the F2 optional system feature. We can also have FSM1 state machine with a set of states (S1), ie, st1 and st2 mandatory states, and st3 optional state, and a set of transitions (T): t1, t2, and t3 encapsulating common and variable behaviors. After that, in the second subphase of domain engineering, we will use as input the scenarios relevant for the stakeholders together with the system architecture, composed of different subsystems and components, and we will identify potential failures both in subsystems and components that can be found in each scenario. Finally, we will integrate the variability information produced in the domain modeling with the SPL design and dependability model elements.

Hazard analysis and risk assessment

The inputs for this activity are a targeted scenario, ie, product/usage context variant, relevant for the stakeholders, along with SPL architectural and behavioral models. After choosing a given scenario earlier identified in the previous step, HARA is carried out aimed at identifying combinations of contributing component failures/faults that may lead to the occurrence of system-level failures named hazards. We can specify hazards by means of logical expressions, such as AND, OR, and NOT operators, involving potential failures in SPL architectural subsystems/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, we analyze interactions among core architectural components to identify potential hazards that can emerge from these interactions. Later, optional, alternative, and mutually inclusive/exclusive architectural components, representing variability defined in the targeted scenario, are analyzed to identify potential hazards that can emerge in such scenarios. This step results in a list of variant-specific hazards.

Considering PLA, a hypothetical aerospace SPL architecture model comprising S1 and S2 mandatory subsystems and a C1 optional component, an omission of S1.out1 AND omission of S2.out1 can cause the Omission of commands hazard in the SC1 scenario. In the next step, still at the system level, we estimate and classify the risk posed by each identified hazard in each scenario using the probabilistic risk tolerability criteria defined in the targeted domain standards. DO-178C3 and SAE ARP 4754A4 aerospace safety considerations establish five software levels, DAL A to DAL E, and prescribe that risk assessment should be performed on the basis of severity and probability levels to classify a system hazard at a given software level. Aerospace MIL-STD-882E38 standard practice for system safety provides a hazard/mishap risk matrix that relates range values for severity and probability attributes with software levels. Thus, engineers estimate the risk posed by a system hazard by assigning values to reliability attributes, eg, severity and likelihood, and later on by checking the corresponding software level in the risk matrix during safety assessment. For example, if the severity of the occurrence of Omission of commands is Catastrophic with probability of occurrence of 10e-3 per hour of operation, then, its risk level is DAL A according to DO-178C and MIL-STD-882E mishap risk matrix.

The risk assigned to a hazard at the system level is inherited by software that contributes to the occurrence of that hazard, to address the targeted software level, eg, DO-178C DAL A. The assignment of software levels to components is modified via architecture. Architectural strategies may contribute to reducing the impact of failures. Thus, different architectural choices may change the assignment of software levels to components. Considering the Omission of commands hazard, the addition of C1 optional component to the architecture may change the software levels assigned to S1 and S2 mandatory subsystems comply with DAL A. This is detailed in the allocation and decomposition of safety requirements, which is also part of the system safety assessment process. The outputs of this activity are a list of identified hazards and their risk classification for each analyzed scenario.
Allocation and decomposition of safety requirements

This activity is part of product line hazard analysis and risk assessment. From the analysis of product line hazards, we specify system and functional safety requirements and allocate Safety Integrity Levels (SILs) for eliminating or minimizing hazard and/or component failure effects on the overall safety. A system safety requirement is the risk reduction measure to mitigate hazard effects. A functional safety requirement relates to design and implementation decisions, stated in the form of system functions, intended for eliminating or minimizing the effects of failures on the overall system dependability. An SIL or safety integrity requirement represents the reliability, eg, in terms of probability and severity attributes, associated with a given component failure/fault, hazard, or functional safety requirement. Reliability attributes may change from one domain-specific standard/consideration to another.

Cross-domain safety standards establish a set of levels of integrity, ranging from most stringent to least stringent, to classify the risk posed by safety threats to the overall system dependability based on the: values assigned to reliability attributes and their position in the standard risk matrix. SAE ARP 4754A and DO-178C aerospace safety considerations define five levels of integrity, named Development Assurance Levels (DALs). DAL A is the most stringent and DAL E the least stringent. Thus, in aerospace safety considerations, we classify the risk posed by a given system hazard or component failure as DAL A, B, C, D, or E, based on the range of the values assigned to severity and probability reliability attributes. The ISO 26262\textsuperscript{2} automotive standard also establishes five levels of integrity, named Automotive Safety Integrity Levels (ASILs), ranging from Quality Management (QM), the least stringent one, to ASIL D, the most stringent, to classify the risk posed by system/component failures. In automotive ISO 26262, we assign an ASIL, to a hazard or a component failure, based on the values assigned to severity, controllability, and level of exposure attributes.

The allocation of SILs to mitigate the effects of hazards or component failures may impact on the system development costs, since achieving a given SIL demands addressing objectives, by performing system engineering activities for producing a set of development artefacts established in the targeted standard/consideration to achieve safety approval according to the level of integrity. The assignment of standard-related objectives to be addressed, activities to be performed, and artefacts to be produced to achieve a given level of integrity (certification level) may change from one domain-specific standard to another.\textsuperscript{39,40}

In this step, we specify system and functional safety requirements and SILs that should be allocated to each identified hazard according to its risk classification. For example, when considering the values assigned to severity and probability of occurrence of Omission of Commands in the given SC1 scenario, as part of the SAE ARP risk assessment process, we should assign DAL A to mitigate the effects of this hazard. On the other hand, the addition of the C1 component to the product architecture in SC2 increases the system reliability, assuming that this component demonstrably share, with other components, the responsibility of keeping the system operating in a healthy state. Thus, a less stringent DAL B is sufficient to mitigate the effects of Omission of commands. It is important to highlight that the addition of a component in the architecture does not necessarily increase the system reliability. This can happen only if the addition is in such a way to demonstrably share the responsibility with other components to addressing a given software level. However, to ensure the safe usage of SPL subsystems and components, which contribute to the occurrence of this hazard, across a set of system variants, we should still assign the more stringent DAL A. Moreover, it is possible to further decompose the SILs allocated to system hazards throughout contributing components and their associated failure modes.

Allocating stringent SILs to mitigate hazard effects may impact on the definition of stringent standard’s objectives and development activities for producing expensive artefacts to achieve safety approval. This can result in increasing both system development and production costs. Higher SILs generally means higher costs, as meeting most stringent objectives typically require more safety measures, and development effort to deliver higher-quality components. SIL decomposition allows a safety-critical system architecture to meet a particular SIL assigned to a system-level failure without all contributing software components having to meet that SIL. For example, if a system failure is raised only when two independent components fail together, these components share the responsibility of meeting the SIL allocated to mitigate the effects of the given system failure.

Prescriptive and process-oriented safety standards and regulations\textsuperscript{2,4} establish rules for decomposing SILs allocated to system failures throughout contributing component failures. In SAE ARP 4754A, if we allocate DAL A to mitigate a system failure, eg, Omission of commands, caused by failures in two independent components\textsuperscript{2} (S1 and S2), we can respectively allocate DAL A and DAL B, or DAL B and DAL A to S1 and S2 subsystems to address DAL A. This probably reduces development effort and costs for achieving safety approval in comparison with addressing DAL A requirements in both subsystems. SIL decomposition allows the efficient allocation of SILs so that we can achieve safety approval in compliance with standards without being unnecessarily stringent or expensive when addressing objectives. We can perform SIL decomposition with the support of automated design optimization algorithms,\textsuperscript{8,41} built upon genetic, penalty-based, and Tabu-Search metaheuristics techniques, after identifying how components can fail and their contribution to the occurrence of hazards during component fault modeling. Rules for decomposing SILs may change from one domain-specific standard to another. Thus, design optimization algorithms available in the literature provide support for decomposing; aerospace DALs,\textsuperscript{41} built upon SAE ARP 4754A\textsuperscript{4} DAL decomposition rules, and automotive ASILs\textsuperscript{8} based on ISO 26262\textsuperscript{2} ASIL decomposition rules.

Variation in the SILs allocated to system hazards can be further propagated throughout the SILs allocated to mitigate the effects of component failures. Additionally, engineers can obtain product line SIL decomposition results, which provides the SILs that should be allocated to ensure the safe usage of SPL components across a range of system variants, from the automated analysis of multiple product-specific SIL decomposition

\*\* Components that are isolated and protected in the architecture.\**
results.6 Performed early in the design, it can support engineers in the specification of a cost-effective SPL development process. The support for decomposing SILs in SPL architectures is outside the scope of this paper.

Finally, the allocation of functional safety requirements aims to identify system functions that should be added to the system architecture for eliminating or minimizing the effects of the occurrence of a hazard or component failure on the overall safety. Redundancy is an example of a functional safety requirement. When a new functional safety requirement is added to the safety-critical product line architecture, we should perform dependability analysis again to evaluate the impact of the newer functionality on overall safety in the context of the targeted scenarios. The outputs of this activity are functional safety requirements, stated in terms of system functions added to the system architecture, and a set of SILs assigned to mitigate the effects of hazards and/or component failures on the overall safety.

Component fault modeling

The inputs for this activity are an SPL architecture model and HARA information. From the analysis of the identified hazards that can be raised in a particular scenario, we make assumptions about how architectural components can fail and contribute to the occurrence of each identified hazard. In this activity, firstly, we choose a particular architectural component and identify potential output deviations, ie, failures in component outputs, that can contribute to the occurrence of each identified hazard. We then specify the potential causes for the occurrence of each identified output deviation by analyzing potential combinations of component internal failures and input deviations. Such analysis continues while there remain architectural components to be analyzed. The component failure/faulty behavior can be specified, using compositional techniques,7-9 as annotations in the SPL architecture model. Such annotations are stated as logical expressions, using AND, OR, and NOT operators, showing how deviations in component output ports can be raised either by component internal failures and/or deviations in the component inputs or combinations of these events. In system safety analysis, component deviations may include unexpected omission of an output or unintended commission of output, or incorrect output values, or output being sent too early or late.8

Considering C1 software component in the SC2 scenario, the causes of an omission of the C1.out1 output can be an omission internal failure of C1 OR the occurrence of an omission failure at C1.in1 OR C1.in2 inputs, ie, Omission-C1.out1 - Omission-C1 OR (Omission-C1.in1 OR Omission-C1.in2) failure expression, stored in the C1-SC2 component fault model library. Local component fault modeling can reflect either real characteristics or simply the design intention for the analyzed architectural components. In both cases, the analysis is useful. At earlier stages, when components are under design and only design intentions are encoded, it is still possible to evaluate the suitability of the proposed design under the encoded intentions of the failure logic, fault propagation, fault mitigation, and fault tolerance of various architectural components. Late in the design, such analysis supports engineers in identifying weaknesses and taking decisions to improve the SPL architecture, eg, by introducing components with improved characteristics or fault tolerant features. For hardware components, where failures are nondeterministic, we specify the failure behavior in terms of values of reliability attributes, eg, failure and repair rates, which might change according to the targeted scenario. For example, we can assign that C2 hardware component failure rate is 10e-7 per hour of operation in the SC1 scenario.

Compositional techniques allow the storing of the component failure behavior in a library, so that other components of the same type can further reuse it. Thus, we can encapsulate variation in component fault behaviors into fault model libraries. Each fault model library contains component failure expressions and failure modes addressing a specific usage scenario. A fault model library may further reuse it. Thus, we can encapsulate variation in component fault behaviors into fault model libraries. Each fault model library contains

Integration of variability information

The inputs for this activity are the SPL feature and context models, along with the SPL architectural, behavioral, and dependability models. This activity extends the variability modeling (application variability modeling) step defined in traditional SPL methods,18-20,42 aimed at establishing mapping links between features and their realization in the architecture model. This is to enable support for variability management in dependability artefacts. In this step, we define mappings linking system and usage context features to their realization in both architectural, behavioral, and dependability model elements. It is important to highlight that mapping links in the application variability model are useful to indicate which versions of components and their associated dependability information should be included in each system variant or in each version of a given variant.

Design and dependability base models can be specified with the support of compositional modeling languages and techniques.5-9 Thus, to select design and dependability model elements and linking them to abstract representations defined in the SPL feature and context models, variant management tools have to communicate with compositional modeling tools. We can manage variability in system models enhanced with dependability information with the support of extensions to existing variant management techniques.11,32,35 eg, BVR toolset.36 This activity encompasses the following tasks: (a) specifying design and dependable-related variation points and their variants from the analysis of interactions between system and usage context features, and scenarios used to guide engineers during dependability analysis; (b) mapping design variants
(defined in the SPL feature model) to their realization in the SPL architectural and behavioral models by defining model elements to be included (replacement) and/or excluded (placement) when each design variant is resolved during the product derivation process; and finally, (c) mapping dependable-related variants (specified in the SPL feature and context models) to their realization in the dependability model by linking each variant scenario considered during dependability analysis to the corresponding information about HARA, the Allocation of Safety Requirements, and failure data (fault model) associated with each variant-specific component.

A mapping link in an orthogonal variability model, which is the modeling technique adopted by the most variant management tools,\textsuperscript{11,32,36} comprises the tuple $M = (V, F, ME)$. $V$ represents a variant, which provides a concrete instantiation of variable domain artefacts, e.g., architectural models, specified in a variation point (i.e., a place in the base model where variation can arise). $F$ is a reference to a feature or a feature set specified in the SPL feature model (variability specification model). Finally, $ME$ is a set of base model elements, e.g., architectural, behavioral and/or dependability model elements, which represent the realization of an SPL system/usage context feature or a feature set ($F$) in the base model.

A mapping link in the BVR\textsuperscript{36} variability model comprises $M' = (V, F, FS \{P_{me}, R_{me}\})$, which is a slight variation of the $M$ tuple. An $FS$ contains references to base model elements to be excluded, named placement elements ($P_{me}$), and to be included, named replacements ($R_{me}$), in a product-specific model during product derivation. Fragment substitution is a BVR toolset concept that refers to base model elements (ME) that should be excluded ($P_{me}$) and included ($R_{me}$) in a variant-specific model when an abstract feature ($F$) from the SPL domain model, representing a variant, is chosen. Thus, an $FS$ establishes a traceability link between an abstract representation specified in the SPL feature model and its implementation/realization in the base model. In our approach, we consider the specification of fragment substitutions ($FS$) with a replacement ($R_{me}$) and an empty placement ($P_{me}$) element in the variability model. Thus, references to base model elements to be included in a product-specific model are stored in replacement elements of mapping links ($M'$) defined in the variability model.

Considering the hypothetical FM feature and CM context models, and product line dependability models (DM) as inputs to create the application variability model (VM), we have $m1$ $(SC1, F1-CX1, FS-SC1-D \{P_{me} \}, R_{me}[HARA-SC1, S1-SC1, S2-SC1])$ mapping linking SC1 variant, which represents the selection of CX1 context and F1 system domain features defined in the SPL feature and context models, to $HARA-SC1$ hazard analysis information, and $S1-SC1$ and $S2-SC1$ component fault model libraries representing the realization of this variant in the dependability model. $F1-CX1$ represents the domain feature associated with $FS-SC1-D$ elements. By using this modeling strategy, there is no need to perform substitutions during the product derivation process. It is important to highlight that a traceability matrix showing the relationships between base model elements and their selection in each possible system variant/configuration, e.g., pure::variants\textsuperscript{11} traceability matrix, can support product line engineers in this activity.

The end of this step yields the enhanced application variability model that links features to their realization in both architecture, behavioral, and dependability model elements. The application variability model of a safety-critical SPL comprises a set of mappings ($m_1, m_2, \ldots, m_n$) linking features, specified in the SPL domain model, to their realization in both design and dependability model elements. The variability information stored in the variability model is useful for variant management tools supporting the automatic/semiautomatic derivation of product-specific design models annotated with dependability information.

### 3.5 DEPendable-SPLE: Application engineering phase

During the application engineering phase, both the SPL design and dependability artefacts, stored in an SPL assets repository following the previous steps, can be reused in the development of system variants that address customer-specific requirements. Thus, from the analysis of the product line feature/context models and customer-specific application requirements, SPL system and dependable-related features are chosen according to the targeted application requirements to derive the product/application feature model (ie, variability resolution or instance model). Additional customer-specific system/product and usage context features, not predicted in the SPL domain modeling and design, can also be added to the application feature model, and to the SPL core assets repository. During the product derivation process, customer-specific design and dependability assets are generated, with the support of a variability management tool, eg. BVR toolset,\textsuperscript{36} from the choices specified in the application/product feature model. Customer-specific system functions can be added to the generated variant-specific design.

The addition of customer-specific system functions and usage contexts may impact on the dependability analysis. Thus, variant-specific hazards and component failures can be identified during application of HARA and component fault modeling, and we can allocate application-specific safety requirements to mitigate their effects on the overall system safety/dependability. Finally, variant-specific fault trees and FMEA results can be generated, with the support of compositional dependability analysis techniques,\textsuperscript{7-9} from the reused HARA information and component fault models. With the exception of fault trees and FMEA synthesis, dependability analysis in the application engineering phase can be seen as iterations of the domain engineering phase. Thus, we only provide a brief description of product hazard analysis and component fault modeling to support the reader in better understanding our approach. We detail each DEPendable-SPLE application engineering approach activity in the following subsections.

#### 3.5.1 Application-specific requirements

**Product-specific requirements**

The inputs for this activity are the product line feature and context models and application requirements. In this activity, design and dependable-related variants defined in the SPL feature and context models that address application requirements are chosen, and in some cases, application-specific
requirements, not provided by the SPL, are added to the product feature and context models. The end of this activity yields product feature and context models. It is important to highlight that the addition of product-specific features may impact on both product design (Figure 2 (iv) - PL architectural customization) and dependability analysis (Figures 2 (iii), (iv), and (v)).

**Identification of candidate scenarios for product HARA**

The product feature/context models, product/application architectural and behavioral models, and the analysts’ domain knowledge are inputs for this activity. After product derivation (ie, PL architectural customization), we perform an analysis of interactions among chosen SPL system and usage context features, application-specific system and usage contexts, and application-specific functions added to the reused product architectural and behavioral models after product derivation. Such analysis is focused on identifying potential scenarios, not predicted in the domain engineering phase, that may impact on the overall system safety/dependability, eg, with the emergence of application-specific system hazards and/or component failures. These scenarios can be used to guide engineers during product hazard analysis and component fault modeling activities by following the same steps defined in the domain engineering phase. It is important to highlight that the addition of application-specific system functions to the SPL design (architectural and behavioral models) can result in the emergence of different scenarios that may impact on the overall safety. These scenarios should be considered for guiding engineers through iterations of dependability analysis activities in the SPL domain engineering phase.

The output of this activity is a set of application-specific scenarios that may impact on overall safety. Emergent application-specific scenarios may cause disruption to the previous assumptions about hazards and component failures made by engineers in the domain engineering phase. There is little that can be done in these cases beyond storing emergent assumptions about hazards, component faults, and safety requirements in the SPL core assets, together with previous assumptions about dependability.

**3.5.2 Product line architecture customization**

**Subsystem and component selection**

The SPL feature and context models, the enhanced SPL variability model with mapping links between features and their realization in HARA information and component failure data, the product feature model, and SPL assets, in this case, architectural, behavioral, and dependability models are inputs for this activity. The purpose of this activity is resolving, with the support of extensions of existing variability management tools, eg, the BVR toolset, the variability specified in SPL architectural, behavioral, and dependability base models, and thus, deriving a product/variant-specific system model with dependability information. After deriving a product system model, functions intended to address application-specific requirements (stated in the product feature and context models) not provided by the SPL architecture can be added to the reused product design. These functions can be specified in the form of subsystems and/or components. The output of this activity is a product system model with dependability information that is in some cases enhanced with application-specific system functions.

The addition of specific functions to the reused product design can result in different scenarios emerging that may impact on the product dependability. These scenarios should be considered by engineers during product dependability analysis. Application-specific functions can also provide feedback to the SPL processes, enhancing the SPL architectural and/or behavioral models. The addition of application-specific system functions to the SPL design core assets may cause the emergence of different scenarios that may impact on the overall system dependability. These scenarios should be considered further by engineers in iterations of dependability analyses in the SPL domain engineering phase. Application-specific scenarios should be identified after the selection of SPL subsystems and components and architectural customization.

**Product hazard analysis**

Application-specific scenarios, product feature and usage context models, product architectural and behavioral models, and analyst's domain knowledge are inputs for product hazard analysis. This activity should be performed by following the same steps defined for HARA in the domain engineering phase. Thus, we should perform product HARA for each identified application-specific scenario, which comprises application-specific product and usage context feature interactions, to identify combinations among component failures that may lead to the occurrence of system-level failures, which were not earlier identified in the domain engineering phase. Next, the risk for the overall safety posed by each application-specific hazard, in different usage scenarios, is estimated, based on the risk tolerability criteria defined in the targeted safety standard/considerations, eg, SAE ARP 4754A severity and likelihood. The output of this activity is a list of product/application-specific hazards and their associated risk classification. Application-specific HARA information can be added to the SPL repository to be further available for reuse in the development (derivation) of other system variants from SPL core assets.

**Allocation of product safety requirements**

Application-specific scenarios and HARA information, along with analysts’ domain knowledge, are inputs for this activity. Thus, from the analysis of product HARA, functional safety requirements, eg, redundancy, can be added to a variant-specific architecture model, and SILs can be allocated to eliminate or minimize hazard effects on overall safety. In the same way as in the domain engineering phase, SILs should be allocated to each product-specific hazard according to their risk classification. A set of application-specific functional safety requirements, ie, system functions aiming at mitigating system-level failures, and SILs allocated to system hazards are the outputs of this activity. Additionally, SILs allocated to
application-specific hazards are further decomposed to contributing component failure modes. SIL decomposition can be performed for a particular system variant with automated support of SIL decomposition genetic and metaheuristic algorithms.8,41

The decomposition of SILs, allocated to application-specific hazards, to contributing components and their failure modes early in the design, can support engineers on the specification of cost-effective application/product development processes according to the level of integrity assigned to subsystem components. Thus, the development process of two different variant-specific subsystem components with different levels of integrity do not need to address the requirements posed by the most stringent level of integrity assigned to a given component. Application-specific functions and allocated safety integrity levels can be added to the SPL repository to refine the SPL design and ensuring the safe operation of components across a range of different system variants not predicted in the product line design.

Product component fault modeling

The inputs for this activity are application-specific scenarios and HARA information. From the analysis of the identified hazards that can be raised in a specific system variant and its associated usage context, assumptions can be made about how application-specific architectural components, ie, reused and application-specific assets, can fail and contribute to the occurrence of each emergent hazard (identified during application HARA). In this step, we analyze each application-specific component to identify potential output deviations that can contribute to the occurrence of each identified application-specific hazard, and their causes, ie, combinations among component internal failures and input deviations, in the same way as in the domain engineering phase. Such analysis continues while there remain application-specific components. At the end of product component fault modeling, we obtain a set of application-specific component fault models, and the reused domain component fault models are enhanced/updated with product-specific dependability information. Thus, the outputs of Product Dependability Analysis are enhanced application-specific design and dependability models and, in some cases, feedback to the safety-critical SPLE processes. Thus, enhancing the SPL architectural, behavioral, and dependability (ie, HARA information and component fault models) models stored in the DEPendable-SPLE repository (see Figure 2).

3.5.3 Product testing

Fault trees and FMEA synthesis

The reused product-specific architectural and behavioral models, enhanced with specific dependability information, are the inputs for performing the synthesis of fault trees and FMEA results with the support of compositional techniques.7,9 Fault trees and FMEA results can be automatically generated, with the support of compositional techniques, from the reused and product-specific dependability information produced during domain and application engineering phases. The accuracy of the generated product-specific fault trees and FMEA results is dependent on whether both domain and application dependability analysis activities were performed with awareness of the impact of design and usage context variations on overall safety. The end of this activity yields application-specific fault trees and FMEA results that can be used to demonstrate that the architecture of a particular system/product variant addresses safety/dependability requirements.

4 CASE STUDY

This section presents the application of the DEPendable-SPLE approach in the Tiriba Flight Control System and Software Product Line (TFC-SPL). In Section 4.1, we present an overview of the TFC-SPL, with focus on domain modeling and design, without going into details concerning dependability aspects. In Section 4.2, we illustrate the dependability approach steps and integration of variability information in the domain engineering phase. Finally, in Section 4.3, we present dependability steps regarding the application engineering phase.

4.1 Tiriba flight control product line

TFC-SPL is part of the Tiriba UAV avionics SPL (see Section 2), comprising a control subsystem with the following goals: to start the flight mode (ie, which can be direct, stabilized, or autonomous), processing and setup of flight commands, keeping flight conditions, and executing commands sent by the navigation subsystem. Although Tiriba was originally designed in MATLAB Simulink,44 to illustrate the integration of variability management and compositional dependability analysis techniques, TFC-SPL design and dependability models were specified in OSATE AADL (Architectural Analysis and Description Language) and AADL Error Annex.7 BVR toolset26 and a BVR enabler adapter to OSATE AADL & Error Annex developed by the authors were used to support variability management in both TFC-SPL design and dependability models.

4.1.1 Tiriba product line domain modeling

TFC-SPL was designed following an extractive strategy by analyzing the original Tiriba Flight Control subsystem Simulink model. Figure 3 shows the interconnected TFC-SPL feature and context models. We created TFC-SPL domain models by following Lee and Kang26 and Hartman and Trew28 multiperspective feature modeling approaches and their refinements.29,30 These modeling approaches allow separating the specification of system features from usage context features and a more clear understanding of their relationships.

TFC-SPL system feature model (Figure 3 (left)) comprises the Pilot Mode, the Landing Type, and Navigation Services variation points. Landing Type variation point encapsulates variation on the UAV landing capability, which can be via activation of a Parachute to smooth the impact on the ground
and avoid potential damage to the UAV when landing in an area where a runaway is not available. Autonomous, ie, the UAV is preprogrammed to land in a specific area or via Remote Control by receiving and executing landing commands sent by an human operator from a ground control station.

Navigation Services variation point encapsulates variation on the Tiriba UAV mission behavior. Thus, the UAV can abort a mission segment (Mission Abortion optional feature) or perform a simulation (Entry Segment Sim feature), whenever the UAV starts a new mission segment, to find the most suitable maneuver to switch between two mission segments. The UAV route can be adjusted whenever the aircraft deviates more than a certain threshold (limit) from the planned mission route (Route Correction). The UAV can also return to specific positions where a picture was not taken during the mission when the Route Tracking feature is chosen. Finally, the Pilot Mode variation point contains four pilot mode options/variants: Manual Pilot is mandatory; Assisted Pilot, Autonomous Pilot, and Autopilot are optional. In the Manual Pilot mode, a human operator send commands to the unmanned aircraft vehicle (UAV) from a ground control station.

In the Autopilot mode, the UAV executes a predefined route. The Assisted Pilot mode allows the operator to send commands to the UAV configured with Autopilot mode. The Autonomous Pilot mode allows the UAV to perform actions according to its current environmental conditions captured by pressure sensors. Assisted Pilot, Autonomous Pilot, and Autopilot optional features can be combined in several different ways, allowing seven different flight control system variants.

TFC system/product variants can operate in a range of different usage contexts defined by combining Airspace, Application, and UAV Size variation points and their variants. Figure 3 (right) shows an excerpt of the TFC-SPL context model. A Tiriba flight control system variant can operate in an Uncontrolled or nonsegregated airspace. This refers to the operation of a UAV outside of a Controlled or segregated airspace, which covers airspace dimensions allocated for exclusive use to specific users. A system variant operating in a segregated/controlled airspace must include the Mission Abortion optional system feature, manual, and autonomous pilot feature, along with route tracking and route correction mission controller capabilities. A system variant operating in an uncontrolled airspace must include Entry Segment Simulation feature.

A Tiriba flight control system variant can be deployed into a Small UAV (a UAV weighing less than 5 kg), or in a Light UAV, a UAV with weight ranging from 5 to 20 kg. The Parachute Landing capability is excluded from a system variant when deployed into a Light UAV and operating in an uncontrolled airspace. A Tiriba system variant can be used for civilian applications, eg, detection of crop diseases in agriculture, topographical surveys and traffic monitoring, urban planning, and power transmission line inspection in the environment monitoring category or for defense applications.

4.1.2 Tiriba flight control product line design

Variability specified in the TFC-SPL feature and context models propagates throughout the design. The Tiriba Flight Control (TFC) product line architecture encompasses four subsystems and 14 components, which are composed of 252 model elements and subcomponents. Figure 4 shows an excerpt of the Tiriba Flight Control SysML main block diagram. The flight control mode function, aimed at initiating different flying modes, is represented by the Mode Switcher component. The flight and set commands functions are supported in the architecture by the PWM Decoder, Fail Safe Controller, and Command Switch for takeoff and landing operations. The purpose of the third flight control function is the provision of navigation operations for the UAV in predefined flight conditions and is represented in the TFC architecture by the Autopilot component.

The Mode Switch subsystem component encapsulates the Pilot Mode variation point shown in Figure 3 (left), while the Command Switch component encapsulates the variation point inherent to the source from the pilot commands, eg, manual pilot or autopilot subsystems, sent to the UAV. The Basic Command Processor (BCP) subsystem and its ports represent the realization of the Autonomous Pilot mode feature. The PWM (Pulse-Width Modulation) Decoder component output port connected to Command Switch block represents the realization of the Manual Pilot mode feature in the design. The realization of Autopilot feature in the design is given by Autopilot, Flight Control Mixer (FCM), and Fail Safe Controller (FSC) subsystem components, PWMDecoder.DecoderFlightsControls output port connected to FlightStabilizer and Command Switch model components, and FlightStabilizerAutopilotSettings output port connected to Assisted Mode Switch component. Finally, the Mode Switcher subsystem output ports which are connected to Assisted Mode Switch and Command Switch components represent the realization of the Assisted Pilot mode feature.

TFC-SPL variation points and their variants specified during domain modeling propagate throughout the product line architecture model. Thus, when we choose Manual Pilot and Autonomous Pilot system features (TFC-MAT system variant) and Controlled airspace context feature, Basic Command Processor and PWM Decoder subsystems, their ports (except BCP.Mode and PWMDecoder.PilotCommands output ports) and connections, shown in Figure 4, should be included in a TFC variant-specific architecture model. On the other hand, Assisted Pilot, Autopilot, Autonomous

---

**FIGURE 3** Tiriba feature model (left) and context feature model (right)
Pilot, and Manual Pilot features, ie, the all pilot modes (TFC-ALL) system variant operating in an Uncontrolled airspace, are materialized in the architecture by all Tiriba flight control subsystems, components, their ports, and connections (see Figure 4). Still in the TFC-SPL, the selection of variants from Pilot Mode and Airspace variation points impacts on the derivation of redundant and nonredundant components in a variant-specific architecture model. Finally, architectural variation inherent to TFC-MAT and TFC-ALL system variants along with variation in the usage context are further propagated throughout the Tiriba Mission Controller behavioral model expressed in a finite state machine.

Variability in the selection of Tiriba mission-related features specified in Figure 3 (left) are propagated throughout the Tiriba mission controller state machine shown in Figure 5, which defines the mission controller configuration using a mechanism of state transitions conditioned to state variable values that define variability. Thus, when the Entry Segment Simulation mission-related system/functional feature is chosen, a simulation is performed whenever the UAV starts a new mission segment (see Figure 5 (left)), in order to find out the suitable approach for switching between two mission segments. When the Route Correction (ie, Feather Threshold) functional feature is selected, the UAV route is adjusted whenever the aircraft deviates more than a certain threshold (limit) from the planned mission route. The realization of this feature is expressed by the correction start state transition in the Tiriba FSM from Figure 5 (left). Finally, when the Route Tracking (ie, Failure Handler) feature is selected, the UAV is able to return to specific positions where a picture was not taken during the mission, which is dispatched by the good simulation state transition in Figure 5 (left).
The activation or deactivation of Tiriba mission controller functional features is defined by alternative flows with Boolean conditional variables (see Figure 5 (right)), which allow enabling or disabling FSM flows and states according to the feature selection. Thus, for each one of the aforementioned mission-related features, there is a variable whose value defines which FSM states and transitions will take place in the final product.

The SimEnableBoolean variable controls the activation/deactivation of the behaviour related to the Entry Segment Simulation feature. When this feature is selected, i.e., when the TFC-ALL (all pilot modes) system variant and uncontrolled airspace context variant are chosen, the SimEnable variable is set true (see Figure 5 (right)). On the other hand, SimEnable is set false when the Entry Segment Simulation feature is not selected. The same is valid for selection/activation and deselection/deactivation of the Route Corrrection and Route Tracking functional features.

It is important to highlight that usage context features may have influence in the selection of system/design features that impact on the Finite State Machines, changing the system behavior. Thus, when uncontrolled airspace usage context and TFC-ALL system variants are chosen, the SimEnable state variable from Figure 5 (right) is set true, activating the transition to the Simulating state (see Figure 5 (left)) in a variant-specific mission controller FSM. When the controlled airspace context and TFC-MAT (manual and autonomous pilot) system variants are selected, FeatherThrHandler and FailureHandler state variables are set true, activating both the correction start and good simulation state transitions illustrated in Figure 5 (left).

4.2 TFC-SPL: Dependability in domain engineering

In this phase, we initially analyzed interactions between Tiriba system and usage context features to identify scenarios to guide dependability analysis. On the basis of the identified scenarios, we performed TFC-SPL hazard analysis and component fault modeling. Finally, we linked Tiriba domain variants to their realization in both Tiriba SPL design and dependability model elements in the Tiriba application variability model.

4.2.1 Identification of scenarios and Tiriba hazard analysis

From the analysis of the TFC-SPL system and usage context variants specified in the feature and context models shown in Figure 3, a range of candidate scenarios can be derived. The composition of pilot mode product variants and usage context variants leads to 84 different TFC system variants. However, since it would be prohibitive performing dependability analysis covering all these variants, only Tiriba Flight Control manual and autonomous (TFC-MAT) and all pilot modes (TFC-ALL) system variants, respectively operating in Controlled and Uncontrolled airspace usage contexts, together with the analyst's domain knowledge, were considered to perform Tiriba dependability analysis. We performed TFC-SPL HARA with the support of domain experts, by considering TFC-MAT and TFC-ALL system variants and their usage contexts, which were analyzed from the perspective of the SAE ARP 4754A4 Risk Assessment process.

Table 1 shows an excerpt of TFC-SPL hazard analysis and risk assessment. Variation in the causes of the No pilot commands and Value pilot commands hazards, and in the risks that they pose for overall safety, were identified in both of the two aforementioned system variants (see Table 1). The causes for the no pilot commands hazard are omission failures in FailSafeController (FSC) and PWMDecoder component outputs when the TFC-MAT system variant and controlled airspace usage context are chosen. On the other hand, omission failures in the FSC, ModeSwitcher, and PWMDecoder component outputs are the causes for the occurrence of the same hazard when the TFC-ALL system variant and uncontrolled airspace usage context are selected. Such variation can also be found in the causes of incorrect value of pilot commands, i.e., value pilot commands hazard, which may change according to the targeted system and usage context variants (see Table 1). Variation in the TFC-SPL hazard analysis further propagates throughout the risk assessment.

In the TFC-SPL risk assessment, different values for likelihood (probability of occurrence) and severity reliability attributes were assigned to classify the risk posed by the no pilot commands and value pilot commands hazards, according to the targeted system and usage context variants. Thus, the probability of occurrence of both omission of pilot commands (i.e., no pilot commands hazard) and incorrect pilot commands (i.e., value pilot commands) are respectively 10e-9 and 10e-7 per hour of operation with catastrophic and hazardous severity when the TFC-MAT system and controlled airspace usage context variants are chosen. On the other hand, the probability of occurrence of both hazards are respectively 10e-7 and

| System Variant | Hazard Analysis | Risk Assessment and Allocation of Safety Req. |
|----------------|----------------|----------------------------------------------|
| TFC-MAT/Controlled | No pilot commands | Omission-FSC.filteredControls AND Omission-PWMDecoder.flightControls | Catastrophic 10e-9 A |
| TFC-ALL/Uncontrolled | No pilot commands | Omission-FSC.filteredControls AND Omission-PWMDecoder.flightControls | Hazardous 10e-7 B |
| TFC-MAT/Controlled | Incorrect pilot commands | Value-FSC.filteredControls AND Omission-ModeSwitcher.controlMode | Hazardous 10e-7 B |
| TFC-ALL/Uncontrolled | Incorrect pilot commands | Value-FSC.filteredControls AND Value-PWMDecoder.flightControls | Hazardous 10e-5 C |
Tiriba control system architecture should be adopted when the Redundant requirements, expressed in terms of architectural decisions, according to the chosen Tiriba flight control allocated to mitigate Tiriba hazard effects. Following the analysis of the TFC-SPL risk assessment, we allocated different functional safety degrees of stringency were assigned to functions and decomposing them throughout the items. Rules allow downgrading the DALs assigned to functions (example, \( \text{DAL A} \)) and their associated failure modes identified during component fault modeling. We performed DAL decomposition for each Tiriba system variant with the automated support of metaheuristic optimization algorithms. These algorithms implement aerospace SAE ARP 4754A rules for allocating DALs at the component level.

Variation in the DALs allocated to mitigate hazard effects in both Tiriba system variants has a direct impact on the development processes to be enacted to comply\(^{11}\) with safety standards/considerations in order to achieve the safety certification\(^2\) of an individual system variant. Thus, expensive objectives, eg, performing the aircraft/system functional hazard assessment with independence,\(^{22}\) prescribed by SAE ARP 4754A to achieve DAL A, should be enacted to mitigate the effects of the occurrence of the value pilot commands hazard when the TFC-MAT system variant operating in a controlled airspace. On the other hand, DAL C is sufficient to mitigate the effects of this hazard when the TFC-ALL system variant operating in an uncontrolled airspace is chosen.

Table 2 shows the decomposition of DALs allocated to the No pilot commands and Value pilot commands hazards (see Table 1) to the Barometric Processor (BP) and PWM Decoder components and their failure modes in both TFC-MAT and TFC-ALL system variants. DALs with different degrees of stringency were assigned to mitigate the effects of different failure modes of a component in the same system variant. For example, DAL A and DAL E were respectively assigned to mitigate the effects of \( \text{OFailure1} \) (omission) and \( \text{VFailure1} \) (value) failure modes on the overall dependability of PWM Decoder subsystem in the TFC-MAT system variant (see column FM-DAL: MAT from Table 2). Such variation may impact on the DAL that should be assigned to ensure the safe use of a given SPL component in a particular system variant, eg, TFC-MAT, by analyzing the DALs allocated to each failure mode of the given component. The most stringent DAL allocated to a failure mode of a component is the required safety requirement to guarantee its safe use in the given system variant. Considering DAL A and DAL E respectively allocated to mitigate the effects of the PWM Decoder omission and value failures in the TFC-MAT system variant, DAL A should be assigned to ensure the safe use of PWM Decoder component in this variant (see column DAL:MAT from Table 2). On the other hand, a less stringent DAL C is sufficient to mitigate the effects of the occurrence of failures in the Barometric Processor subsystem in the TFC-MAT system variant.

Variation in the DALs assigned to different Tiriba SPL components in the same system variant may impact on the structure of development processes to be enacted to ensure the safety of a component in the given system variant. Different objectives should be addressed and different

| Component          | Failure Mode | FM-DAL: MAT | DAL: MAT | FM-DAL: ALL | DAL: ALL | SPL: DAL |
|--------------------|--------------|-------------|----------|-------------|----------|----------|
| Barometric Processor | OFailure1    | C           | C        | C           | C        | C        |
|                     | OFailure2    | C           | C        | C           | C        | C        |
| PWM Decoder         | OFailure1    | A           | A        | C           | C        | A        |
|                     | OFailure2    | -           | A        | C           | C        | A        |
|                     | VFailure1    | E           | C        | C           | C        | C        |
|                     | VFailure2    | -           | C        | C           | C        | C        |

- : failure mode is absent in the component in a particular system variant.

\(10^{-5}\) per hour of operation with hazardous and major severity when the TFC-ALL system variant and uncontrolled airspace are chosen. Variation in the assignment of values to hazard severity and likelihood may impact on the allocation of functional safety requirements and allocation and decomposition of DALs to avoid or minimize hazard and/or component failure effects on the overall safety of TFC-SPL components.

### 4.2.2 Allocation and decomposition of safety requirements in the Tiriba SPL

After estimating the risk posed by each identified hazard, functional safety requirements, and safety integrity requirements, in terms of DALs, were allocated to mitigate Tiriba hazard effects. Following the analysis of the TFC-SPL risk assessment, we allocated different functional safety requirements, expressed in terms of architectural decisions, according to the chosen Tiriba flight control system and usage context variants. A Redundant Tiriba control system architecture should be adopted when the controlled airspace usage context is chosen. On the other hand, a nonredundant architecture is sufficient to address the safety requirements when the Tiriba UAV is intended to operate in an uncontrolled airspace.\(^{43}\)

Based on the analysis of the risk assessment (Table 1), DALs with different degrees of stringency were also assigned to mitigate hazard effects in different TFC system variants (see column DAL in Table 1). Thus, we assigned DAL B to mitigate the effects of the occurrence of the value pilot commands hazard in the TFC-MAT system variant operating in a controlled airspace. On the other hand, DAL C is sufficient to mitigate the effects of this hazard when the TFC-ALL system variant operating in an uncontrolled airspace is chosen.

Compliance checks are only sampling events. For example, compliance to DO-178C safety considerations for airborne systems is only to verify if the implementation of a given set of requirements is correct. This is important because if there is a fundamental flaw in the safety assessment process, safety requirements may not be complete and correct, thus, leading to a faulty system even at DAL A.\(^{23}\)

\(^{11}\) Compliance checks are only sampling events. For example, compliance to DO-178C safety considerations for airborne systems is only to verify if the implementation of a given set of requirements is correct. This is important because if there is a fundamental flaw in the safety assessment process, safety requirements may not be complete and correct, thus, leading to a faulty system even at DAL A.

\(^{22}\) Independence: separation of responsibilities, which ensures the accomplishment of objective evaluation. For software verification activities, eg, hazard analysis, independence is achieved when the activity is performed by a person(s) other than the developer of the item under analysis.\(^{3}\)
activities should be performed to produce artefacts to ensure the safety of the different components in a given system variant. By analyzing the DALs allocated to mitigate component failures in the TFC-MAT system variant (see Table 2), it can be seen that the PWM Decoder is a highly critical component with DAL A. Thus, expensive safety objectives with independence, eg, SAE ARP 4754A²: aircraft functional hazard assessment, and DO-178C²: verification of additional code, should be addressed. On the other hand, less critical components, eg, the Barometric Processor a DAL C component, demand less stringent safety objectives that do not require independence.

The definition of development processes according to the targeted level of the integrity of each TFC-MAT system component help to avoid unnecessary costs for achieving compliance with standards without compromising dependability/safety. In this way, considering TFC-MAT components and the DO-178C safety standard, all DO-178C objectives (66) were only assigned to ensure the safety of the DAL A PWM Decoder subsystem. On the other hand, the assignment of 57 DO-178C objectives were sufficient to ensure the safety of the DAL C Barometric Processor subsystem instead of allocating the remaining DAL B and DAL A objectives, which would increase the development costs of the Barometric Processor subsystem in the TFC-MAT system variant.

Variation in the DALs allocated to the Tiriba flight control hazards in different variants propagates to the DALs allocated to mitigate the effects of contributing component failure modes. DALs with different degrees of stringency were assigned to mitigate the effects of the PWM Decoder subsystem failure modes according to the targeted system and usage context variants. DAL C is sufficient to mitigate the effects of the occurrence of Failure 1 failure mode on the overall safety of the PWM Decoder in the TFC-ALL system variant. On the other hand, we assigned a stringent DAL A to mitigate the effects of the occurrence of this failure mode in the TFC-MAT system variant. Thus, the DAL allocated to a given component failure may change according to the targeted system variant. Such variation is further propagated to the assignment of DALs to the PWM Decoder component in different system variants. Thus, a stringent DAL A was assigned to ensure the safety of PWM Decoder component in the TFC-MAT system variant (see column DAL:MAT from Table 2). On the other hand, a less stringent DAL C is sufficient to mitigate PWM Decoder failure effects in the TFC-ALL variant (see column DAL:ALL from Table 2). Such variation may impact on the structure of development processes to be enacted to ensure the safety of the PWM Decoder component in different system variants. Consequently, engineers must address different objectives, performing different activities for producing artefacts to achieve the PWM Decoder component safety approval in each TFC system variant.

The analysis of the DALs assigned to the Tiriba SPL components in different system variants may support engineers in identifying the safety requirements that should be allocated to ensure the safe use of components across the SPL, ie, across a set of different system variants relevant for the stakeholders. We perform such analysis on the basis of the following principle: The most stringent DAL assigned to a given failure mode of a Tiriba SPL component across multiple system variants is the DAL that should be assigned to ensure the safe use of a component across the analyzed system variants, in this case, TFC-MAT and TFC-ALL. Thus, from the analysis of the DALs assigned to the PWM Decoder component in different Tiriba system variants, ie, DAL A in TFC-MAT and DAL C in TFC-ALL, it is possible to infer that DAL A should be assigned to this component to ensure its safe use across the SPL.

Although the most stringent DAL A was assigned to the PWM Decoder subsystem in only one TFC variant, and less stringent DAL C on other variant, DAL A was assigned to that component to avoid loss and unnecessary budget, ie, additional re-verification, validation and testing effort, in case the PWM Decoder is further used in most expensive variants. From such analysis, we obtain the decomposition of DALs assigned to TFC hazards across the SPL. Early in the design, it can support engineers on structuring a cost-effective development process for the Tiriba Flight Control product line. A detailed discussion about allocation and decomposition of safety requirements in safety-critical software product line design can be found elsewhere.

### 4.2.3 Tiriba SPL component fault modeling

In the TFC-SPL, different component failures may contribute to the occurrence of each identified hazard in different TFC variants and usage contexts. During the Tiriba component fault modeling, we specified 106 failure expressions, stored in fault model libraries, which were added to 47 Tiriba flight control model elements.

Table 3 shows an excerpt of the Basic Command Processor (BCP) component fault models. Different output deviations may occur in this component in different Tiriba system variants. Omission of both the AutopilotSettings and Mode component’s output ports may be raised in the BCP component, contributing to the occurrence of hazards, when the TFC-ALL system variant operating in an uncontrolled airspace is chosen (see column Output Deviation from Table 3). On the other hand, only an omission of the AutopilotSettingsOutputport contributes to the occurrence of hazards when the TFC-MAT system variant operating in a controlled airspace is chosen. Additionally, different combinations among component input deviations and internal failures contribute to the occurrence of Omission-AutopilotSettings output values in both TFC-MAT and TFC-ALL system variants (see column Failure Expression in Table 3).

We specified the causes of an output deviation in a failure expression using logical operators (AND, OR, NOT) describing how combinations of the internal and input failures of a component may lead to the occurrence of an output deviation. The occurrence of an internal omission failure in the BCP component OR omission failures in both the BCP inputs can raise an Omission-AutopilotSettings output deviation in the TFC-ALL variant. Conversely, an internal omission failure in the BCP OR the occurrence of an omission failure in one of the BCP inputs may raise an Omission-AutopilotSettings in the TFC-MAT system variant. Thus, when variability in the dependability assets is solved for the TFC-ALL system variant, only the component fault model associated with this variant is included in a variant-specific dependability model. Such variation is then propagated throughout variant-specific fault trees and FMEA artefacts. In order to support the systematic reuse of the TFC-SPL dependability
TABLE 3  An excerpt of Tiriba component fault analysis and modeling

| System Variant | Component | Output Deviation                  | Failure Expression                                      |
|---------------|-----------|-----------------------------------|---------------------------------------------------------|
| TFC-MAT/      | BCP       | Omission-AutopilotSettings        | OFailure1 OR (Omission-BasicCommand OR Omission-SensorData) |
| Controlled    |           |                                   |                                                         |
| TFC-ALL/      | BCP       | Omission-AutopilotSettings        | OFailure1 OR (Omission-BasicCommand AND Omission-SensorData) |
| Uncontrolled  |           |                                   |                                                         |
|               |           | Omission-Mode                     | OFailure2 OR (Omission-BasicCommand AND Omission-SensorData) |

TABLE 4  Pilot mode/usage context variants and their realization in the Tiriba dependability model

| Variation Point | Fragment Substitution | Fragment Type | HARA Information | Component Failure Data |
|-----------------|-----------------------|---------------|------------------|------------------------|
| TFC-ALL/        | Replacement           | ALL-NoPilotCommands | ALL-PWD-Fault-Model |
| Uncontrolled    |           | ALL-ValuePilotCommands | ALL-MSW-Fault-Model |
| Pilot Mode/     | Replacement           | MAT-NoPilotCommands | MAT-PWD-Fault-Model |
| Airspace        |           | MAT-ValuePilotCommands | MAT-BCP-Fault-Model |
| TFC-MAT/        | Replacement           | MAT-NoPilotCommands | MAT-MSW-Fault-Model |
| Controlled      |           | MAT-ValuePilotCommands |                       |

assets, ie, HARA information and component fault models, we manage the impact of design and usage context variations on dependability information during variability modeling.

4.2.4 | Integration of dependability information into TFC-SPL variability model

We specified the TFC-SPL domain feature and context models, and the application variability model with the support of the BVR variability management toolset, and AADL Error Annex enabler adapter for the BVR toolset developed by the authors. Whereas variability management techniques available in the literature do not provide native support for managing variability in dependability base models (ie, system models enhanced with dependability information), in this paper, the authors built an adapter to the BVR toolset for establishing a link between the BVR realization editor and OSATE AADL model editors. This enables BVR for supporting variability management in AADL design base models enhanced with dependability information produced using OSATE compositional modeling toolset.7

The AADL BVR enabler adapter extends OSATE AADL and Error Annex model editors to enable the BVR tool to communicate with OSATE-based model editors. This is needed for allowing BVR to manage variability in both AADL structural/architectural, behavioral, and dependability base models. Since the BVR toolset is built upon the Eclipse Modeling Framework platform, the adapter was implemented as an Eclipse-based plugin. A discussion about the AADL enabler adapter for BVR tool developed by the authors is outside the scope of this paper. Details about how to create an adapter to the BVR tool and configuring a BVR variability model for a particular software product line can be found elsewhere.36 We have defined BVR fragment substitutions (FS) (see Section 3.4.2) to show how variability in the TFC product line AADL design and error models are resolved when TFC-MAT/Controlled and TFC-ALL/Uncontrolled variants are chosen.

Table 4 illustrates BVR replacement elements, and fragment substitutions (ie, mapping links) associated with TFC-MAT/Controlled and TFC-ALL/Uncontrolled system/context variants specified in the BVR variability model for the TFC product line. Each fragment substitution contains an empty placement element, and a replacement element with references to the design and dependability base model elements that should be included in a Tiriba variant-specific system model when a domain variant is chosen. The table columns respectively represent Variation Point (ie, a variation point specified in the SPL domain model), Fragment Substitution (ie, a variant associated with a variation point), Fragment Type (in this case only replacements), HARA Information, and Component Failure Data.

Table 4 shows an excerpt of mappings linking dependable-related domain variants to their realization into dependability model elements, defined in the TFC-SPL variability model. Thus, when the TFC-MAT system variant operating in a controlled airspace usage context is chosen, HARA information and component fault models associated with other TFC system variants are removed from the dependability model. On the other hand, as specified in the replacement element,94 HARA information and component fault models associated with the TFC-MAT system variant, eg, the

Available on-line: https://github.com/aloliveira/aadl-error-annex-and-bvr
The TFC-SPL variability model comprises eight fragment substitutions, one empty placement, and eight replacement elements. We defined four fragment substitutions to specify the realization of system features in both the Tiriba SPL architectural and behavioral models, and four fragment substitutions were defined to specify the realization of dependability-related and usage context features in the SPL dependability model specified as dependability annotations in the TFC-SPL AADL design.

4.3 TFC-SPL: Application engineering

In this phase, we defined application feature and context models for two different Tiriba system variants. We input these models to the BVR toolset supporting the derivation of variant-specific system models enhanced with dependability information. We performed iterations of dependability analysis steps defined in the domain engineering phase for identifying application-specific scenarios that may impact on system dependability, along with potential hazards and component failures that can emerge from these scenarios.

4.3.1 Tiriba product requirements engineering and architecture customization

We performed product requirements engineering from requirements for agriculture and environment monitoring applications. For each application, we have chosen the Tiriba flight control system and usage context features (see Figure 3) that addresses its requirements. Thus, we have chosen manual, autonomous, assisted, and automatic pilot system features, and the uncontrolled airspace context feature that addresses agriculture application requirements (TFC-ALL system variant). We also selected manual and autonomous pilot system features and the controlled airspace context feature to address environment monitoring application requirements (TFC-MAT system variant). During the specification of feature and context models for these two applications, we identified Weather Conditions variant-specific usage context features such as rainy, stormy, wind, and visibility, which can be cloud or sunny that may impact on the system dependability properties. Since weather conditions features may impact on dependability properties of other system variants, we updated the Tiriba product line context model with these features (see Figure 3 (right)). The addition of these features may impact on both the Tiriba application design and dependability analysis. The addition of an application-specific function, eg, redundant autonomous pilot, not available in the Tiriba SPL core assets, to a system variant may also impact on design and dependability properties.

After specifying feature and context models for agriculture (TFC-ALL) and environment monitoring (TFC-MAT) applications, by choosing system and usage context features defined in the Tiriba SPL domain model, we started the product derivation process. For each TFC system variant, the following artefacts were input to the BVR toolset: the TFC-SPL feature and context models, the product feature and context models, the TFC-SPL variability model, and Tiriba Flight Control AADL design enhanced with AADL Error Annex dependability annotations.7 Thus, we obtained TFC-MAT and TFC-ALL variant-specific AADL design and error models. Variability management in the TFC-SPL enabled the systematic reuse of almost 80% of HARA and component fault modeling dependability information, produced during the domain engineering phase, in the derivation of each Tiriba system variant.

4.3.2 Tiriba product hazard analysis and component fault modeling

By combining rainy and stormyweather condition usage context features added to the TFC-MAT system variant, newer scenarios, not predicted in the domain engineering phase, that may impact on system dependability properties can emerge. Thus, we have considered the TFC-MAT system variant operating under rainy and stormy weather conditions to perform product hazard analysis and component fault modeling. We did this to identify combinations among component failures, which were not earlier identified in the domain engineering phase that may lead to the occurrence of system and component-level failures.

We estimated the risk for overall safety posed by each identified product-specific hazard in different scenarios, based on severity and probability risk tolerability criteria defined in SAE ARP 4754A. Tiriba product hazard analysis and component fault modeling tables are available elsewhere.46 During TFC-MAT product hazard analysis, different hazards, with different causes and risks posed for system dependability, were identified in the aforementioned scenarios.

A delay in receiving pilot commands hazard, with catastrophic severity, and low likelihood (ie, 10e-9 per hour of operation), can emerge due the occurrence of late failures in both the FSC and PWM Decoder component's outputs in the TFC-MAT system variant operating under stormy weather conditions. On the other hand, when the TFC-MAT system variant is operating under rainy weather conditions, the occurrence of an early failure in the FSCfilteredControls output OR a commission failure in the PWMDecoderflightControls output can lead to the occurrence of reception of pilot commands earlier than intended hazard, with a hazardous severity and probability of occurrence of 10e-7 per hour of operation. From the analysis of risk classification associated with TFC-MAT product-specific hazards, we assigned DAL A to mitigate effects of the occurrence of a delay in receiving pilot commands and DAL B to minimize the effects of the occurrence of receiving pilot commands earlier than intended.

The emergence of two additional hazards in the TFC-MAT system variant operating in a controlled airspace under stormy and rainy weather conditions may impact on the structure of reused component fault models. Thus, additional component failure modes that contribute to the
occurrence of product-specific hazards can be raised. Therefore, we enhanced the Tiriba BCP component fault model\textsuperscript{22} with additional output deviations that may contribute to the occurrence of two additional TFC-MAT application-specific hazards identified during product hazard analysis.

TFC-MAT product-specific HARA information and component fault models can be added to the Tiriba SPL repository to be further available for reuse in the development (derivation) of other system variants from Tiriba SPL core assets. TFC-MAT product-specific hazards and component failure data, identified during iterations of the hazard analysis and component fault modeling in the application engineering phase, may lead to different failure/fault propagation not predicted during the TFC-SPL dependability analysis. Thus, different fault trees, with different fault propagation, and FMEA results with different component failure effects on the overall safety can be generated for a particular TFC system variant.

4.3.3 Synthesis of fault trees and FMEA results for Tiriba system variants

Product/variant-specific TFC AADL design and error models were input to generate fault trees for 10 product-specific hazards (for four different TFC system variants), and FMEA results were synthesized from the generated fault trees. Figure 6 shows excerpts of the No pilot commands fault trees generated for the TFC-MAT and TFC-ALL system variants. The No pilot commands fault trees generated for two different system variants show the impact of SPL variation on hazard causes. Thus, the occurrence of an omission of PWMDe- coder.flightControls\textnumero AND FailSafeController.filteredControls component’s outputs leads to the occurrence of the absence of pilot commands system failure when TFC-MAT system variant is chosen (see Figure 6 (left)). On the other hand, the occurrence of an omission of ModeSwitcher.controlMode component output failure in conjunction with the two aforementioned output failures are the causes of the absence of pilot commands when TFC-ALL system variant is chosen (see Figure 6 (right)).

Variation in the No pilot commands fault tree gates propagates throughout the number of failure cut sets. A failure cut set is a combination of basic events that can cause the top event of a fault tree (ie, a system-level failure).\textsuperscript{17} We found an example of such variation in the number of failure cut sets that contribute to the occurrence of an omission of pilot commands system failure in the Tiriba No pilot commands fault tree, which may change according to the chosen system variant. Thus, 60 failure cut sets may contribute to the occurrence of an omission of pilot commands when TFC-MAT system variant is chosen. On the other hand, 108 failure cut sets may contribute to the occurrence of the same top event from No pilot commands fault tree when the TFC-ALL system variant is chosen. Such a difference is justified by the increase in the number of components and failure modes that contribute to the occurrence of No pilot commands fault tree top event in the TFC-ALL system variant.

Variation in Tiriba product fault tree gates, nodes, and failure cut sets propagates to the Failure Modes and Effects Analysis\textsuperscript{46} (FMEA). Thus, failure modes that directly or indirectly contribute to the occurrence of the No pilot commands hazard might change according to the targeted Tiriba system variant. In the FMEA results for Tiriba system variants,\textsuperscript{46} different PWM Decoder component failures may directly or indirectly contribute to the occurrence of Tiriba hazards (see Table 1) in each system variant.

Tiriba product FMEA results\textsuperscript{46} show the effects of the occurrence of each component failure on overall safety, and whether a given component failure is a single point of failure or not, which may change from one system variant to another. Thus, when the TFC-ALL system variant is chosen, the occurrence of OFailure1 and OFailure2 omission failures indirectly contributes, in conjunction with failures from other components, to the occurrence of omission of pilot commands (ie, no pilot commands) system failure, and the occurrence of VFailure1 and VFailure2 value failures, in
conjunction with failures from other components, indirectly contributes to the occurrence of the incorrect value for pilot commands (ie, value pilot commands) hazard. On the other hand, when TFC-MAT system variant is chosen, only the occurrence of OFailure1 omission failure indirectly contributes, in conjunction with failures from other components, to the occurrence of omission of pilot commands system failure, and the occurrence of the VFailure1 value failure, ie, a single point of failure, directly contributes to the occurrence of the incorrect value of pilot commands hazard.

Finally, considering both Tiriba system variants, 30 component failures contribute to the occurrence of system failures in the TFC-ALL variant. Since the TFC-MAT system variant comprises fewer components in comparison with the TFC-ALL variant, only 24 component failures directly or indirectly contribute to the occurrence of system failures in the TFC-MAT system variant.

5 | DISCUSSION

In this section, we present an analysis of the impact of variation in system and usage context features on both design and dependability analysis. ie, on hazard analysis and risk assessment, allocation and decomposition of safety requirements, and component fault modeling and propagation. We performed such analysis, which is one of the contributions of this paper, by considering the following system architecture models enhanced with dependability information: aircraft braking system, door controller, Tiriba UAV, and automotive braking system software product lines. Their architecture models were specified in AADL and Simulink, and dependability annotations with the support of AADL Error Annex, and HiP-HOPS compositional dependability analysis techniques.

5.1 | The impact of variability on system design

In data-flow oriented architectural models, structural/architectural variability can be found in systems, subsystems, components, their ports, and connections expressed by flows from input to output ports and vice versa, which may change according to the chosen system and usage context features. By considering the TFC-SPL developed with Simulink model blocks, the selection of the TFC-MAT system variant from the Pilot Mode variation point, by choosing Manual and Autonomous Pilot system features shown in Figure 3 (left), drives the selection of the PWM Decoder component and the BCP subsystem, their ports, and connections (see Figure 4) during the product derivation process. It is important to highlight that variation in usage context features combined with variation in system features, and isolated variation in the usage context may also impact on the derivation of variant-specific architectural and behavioral models. Thus, architectural variation inherent to system and usage context variants can be further propagated to system behavior, which can be expressed in a finite state machine.

In safety-critical SPL processes, variation in system and usage context features may also impact on the system behavior. Thus, FSM states, state transitions, and events that trigger state transitions may change according to the selection of the targeted system and usage context features. Variation in the FSM can directly impact on elements of the system architecture, changing data port values, along with the configuration of components and their connections. Variability in a finite state machine can be found in the number and 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/or output data, states, and state transitions. A state may also have different local variables with different values. In addition, an FSM state may be involved in different state transitions according to the targeted product and usage context variants. Variation in a state transition can be found in its source and target states and in the event that triggers the transition. The execution priority order and outgoing events of a state transition may also vary according to the targeted system and usage context variants. A state Transition Event may be triggered by different mode/states with different guard conditions. A guard condition should be satisfied for the transition from a source state to a target state. Transition Events may also have Timing Constraints. For example, a transition event should be dispatched after a system failure on an interval of one second or less than one second, depending on the targeted product variant.

Effects of a state transition event represent changes in state variables, which may vary according to the targeted system and context variants. Finally, variation in the state transitions may impact on the dispatch of different outgoing events affecting both the behavior and structure/architecture of a given system/product variant. Examples of outgoing events are changes on states and state variables from other finite state machines, and changes in structural/architectural model elements, eg, systems, subsystems, subcomponents, with their connections and port values. Thus, different states can be achieved, and different values can be assigned to state variables according to the targeted system and usage context variants.

Variation in finite state machines can be found in the Tiriba optional mission-related features expressed by the activation/deactivation of states and transitions from the Tiriba mission controller state machine (Figure 5) according to values assigned to FSM variables. Thus, when the Entry Segment Simulation mission-related system/functional feature is chosen (see Figure 3 [left]), ie, when all Tiriba pilot mode system features and the uncontrolled airspace usage context feature (see Figure 3) are chosen (TFC-ALL/Uncontrolled variant), the SimEnable state variable is set true (Figure 5 [right]), activating the transition to the Simulating state (see Figure 5 [left]). In addition to the impact of system and usage context feature variations on architectural and behavioral models in conventional SPLs, such variation may be further propagated to dependability analysis in safety-critical SPL development processes.
5.2 | Variability and SPL dependability analysis

Variation in both SPL architectural and behavioral models, defined in system and usage context variants, can be further propagated to the safety life cycle of a safety-critical SPL. This impacts on system Hazard Analysis and Risk Assessment (HARA) (ie, ISO 26262 Part 3-7: HARA, and SAE ARP 4754A Sec. 5.1: Aircraft and System Level FHA), component fault analysis and modeling, and allocation and decomposition of safety requirements (ie, ISO 26262 Part 3-8: Functional-Safety Concept and SAE ARP 4754A Sec. 5.2: DAL Assignment). Variation in HARA can be further propagated to dependability engineering activities, required by safety standards/considerations and certifying authorities, to achieve system safety certification and component approval. These include component fault analysis and modeling, Fault Tree Analysis (FTA) and FMEA, and allocation and decomposition of safety requirements.

The production of safety/dependability artefacts contributes to increases in the development costs of safety-critical systems. Understanding how SPL product/system and usage context variation impact on dependability analyses may help to achieve the systematic reuse of both system design and dependability artefacts, reducing the certification costs of individual system variants. Combinations among system and usage context variants may be useful to derive scenarios, which can be used to guide engineers in performing system dependability analysis/modeling in safety-critical SPL architectures. In a safety-critical SPL, different failure conditions can be raised, leading to different system-level failures (ie., hazards), with different probability, severity, and criticality levels. Different safety requirements, in terms of functions or SILs, can be allocated to avoid or minimize hazard and/or component failure effects on overall safety according to design choices and targeted usage contexts. We present a detailed description of a set of identified variability types that can be found when performing dependability analyses for a safety-critical system and software product line in the following section.

5.2.1 | Variability in hazard analysis and risk assessment

During hazard analysis, different hazards and hazard causes can emerge according to the targeted product/system and usage context variants. Examples of such variation can be found in the causes of the No pilot commands and Value pilot commands hazards that may change according to the selection of Tiriba system variants (see Table 1). Variation in hazard definition and hazard causes, identified during hazard analysis, can be further propagated throughout the risk assessment. In the risk assessment, variation can be found in the values assigned to reliability attributes, eg., SAE ARP 4754A likelihood and severity, used to classify the risk posed by each system hazard for overall safety (see Risk Assessment columns from Table 1). Thus, in the TFC-SPL, the likelihood of an omission of pilot commands is 10⁻⁹ per hour of operation with catastrophic severity when the TFC-MAT system variant operating in a controlled airspace is chosen. On the other hand, the addition of the Mode Switcher and Autopilot components, respectively representing assisted and automatic pilot mode features (see Figure 3 (left)), contributes to increase the system reliability when the TFC-ALL system variant operating in an uncontrolled airspace is chosen. This results in a reduction of the severity of omission of pilot commands to hazardous.

Variability in the assignment of values to reliability attributes, eg., severity, during hazard classification, may impact on the allocation and decomposition of functional safety requirements and/or Safety Integrity Levels (SILs) to avoid or minimize the effects of the occurrence of hazards on overall safety. Thus, different safety requirements can be allocated to mitigate hazard effects according to the targeted system and usage context variants. Still in HARA, after classifying the risk posed by the identified hazards and contributing component failures, variation can be found in the allocation of functional safety requirements and SILs to mitigate the effects of system or component-level failures on overall safety.

An example of variation in the assignment of safety integrity levels can be seen in the allocation of DALs with different degrees of stringency to mitigate the effects of the occurrence of the value pilot commands hazard identified during the Tiriba SPL dependability analysis (see Table 1). DAL B was assigned to mitigate the effects of this hazard in the TFC-MAT system variant operating in a controlled airspace. On the other hand, a less stringent DAL C is sufficient to mitigate the effects of this hazard in the TFC-ALL system variant.

Variation can also be found in functional safety requirements, expressed in terms of architectural decisions that must be taken to eliminate or minimize the effects of system or component failures, which might change according to the targeted system and usage context variants. In the Tiriba product line, the control system architecture should be Redundant when the controlled airspace is chosen. On the other hand, a non-Redundant control system architecture is sufficient to ensure the safety of the Tiriba UAV operating in an uncontrolled airspace.

5.2.2 | Variation and its impact on allocation and decomposition of safety requirements

Variability in the allocation and decomposition of SILs relates to the variation on the mitigation mechanisms for handling the risk posed by a given system hazard, component failure, or component, which might change according to the targeted system and usage context variants. Variation in the SILs allocated to mitigate hazard effects in different system variants may be further propagated to their decomposition throughout component failures, identified during component fault modeling, that directly or indirectly contribute to the occurrence of system hazards.

Table 2 shows an example of variation in the decomposition of safety requirements, stated in terms of DALs, assigned to mitigate the effects of hazards in different variants (see Table 1) throughout the Tiriba Barometric Processor and PWM Decoder components and their failure modes. DAL C is sufficient to mitigate the effects of the occurrence of the Ofailure1 failure on the PWM Decoder subsystem in the TFC-ALL system variant. On the other hand, when TFC-MAT variant is chosen, DAL A should be assigned to mitigate the effects of the occurrence of this failure mode. Variation in the allocation fo SILs to component failures of a given component in the same system variant is further propagated throughout the assignment of SILs to ensure the safe use of the component in a particular system variant.
Considering DAL A and DAL E, respectively assigned to mitigate the effects of omission and value failures in the PWM Decoder component (column FM-DAL-MAT from Table 2), DAL A must be assigned to ensure the safe use of this component in the TFC-MAT variant. Additionally, we can find variation in the SILs assigned to mitigate the failure effects of different components in the same system variant, eg, DAL C assigned to the Barometric Processor, and DAL A to the PWM Decoder in the TFC-MAT variant (see Table 2).

Variation in the allocation of SILs to mitigate failure modes of a component in different system variants can be further propagated to the assignment of SILs to ensure the safety of a given SPL component in different system variants. Considering the PWM Decoder component and its failure modes (Table 2), DAL C is sufficient to mitigate component failure effects in the TFC-ALL variant, and a stringent DAL A is required to ensure the safe use of the PWM Decoder in the TFC-MAT variant.

Variation in the assignment of SILs to mitigate failure effects of a given component in different variants can be further propagated to the assignment of SILs to ensure the safety of an SPL component across different system variants, ie, across the SPL. We can perform an analysis of the safety requirements that should be allocated to ensure the safe use of an SPL across a set of targeted variants on the basis of the following principle: the most stringent safety requirements, expressed in terms of SILs, assigned to a failure mode of a component across multiple system variants must be the SIL to be assigned to ensure its safe use across the SPL. From the analysis of Table 2, a stringent DAL A and a less stringent DAL C must be respectively assigned to ensure the safe use of the PWM Decoder and the Barometric Processor components across Tiriba system variants.

Variability in the allocation and decomposition of SILs may impact on the structure and costs of both system variants and software product line development processes, since prescriptive safety standards/considerations, eg, ISO 26262 for automotive, aerospace DO-178C and SAE ARP 4754A, establish different objectives, system engineering activities, eg, verification, validation, and testing, to be carried out, and artefacts to be produced to achieve these objectives for each level of integrity.

Addressing more stringent SILs demands achieving the most stringent objectives, by performing stringent system engineering activities to produce expensive development artefacts, thus, increasing the development and certification costs. Allocating less stringent SILs to less-critical SPL components and most stringent SILs only to highly critical components can reduce both system's (variant) and SPL development and certification costs. A heuristic that expresses the relative cost jumps of developing critical system components to address different SILs, eg, in terms of DALs, can be used to support engineers in estimating system/SPL development costs according to the SILs allocated to system hazards and component failures.

For example, we can adopt the following cost heuristic to estimate the costs of developing the TFC-SPL: 0 to address DAL E, 10 for DAL D, 20 for DAL C, 40 for DAL B, and 50 for DAL A. We use this cost heuristic for illustrative purposes, but any other that engineers find more suitable can be used instead. By considering this cost heuristic, and where DAL A and DAL C are assigned to ensure the safe use of the PWM Decoder and the Barometric Processor subsystem components across the TFC-SPL (see column DAL-SPL from Table 2), the estimated cost for developing these TFC-SPL components is 70 (20 [DAL C] + 50 [DAL A]), instead of 100 in case of assignment of DAL A to both components. Thus, instead of allocating the most stringent DAL A assigned to a component failure to ensure the dependability of all components in a given usage context (see Table 2), a stringent DAL A and a less stringent DAL C must be respectively assigned to ensure the safe use of the PWM Decoder and the Barometric Processor components across Tiriba system variants.

Variability in the allocation and decomposition of SILs may impact on the structure and costs of both system variants and software product line development processes, since prescriptive safety standards/considerations, eg, ISO 26262 for automotive, aerospace DO-178C and SAE ARP 4754A, establish different objectives, system engineering activities, eg, verification, validation, and testing, to be carried out, and artefacts to be produced to achieve these objectives for each level of integrity.

Addressing more stringent SILs demands achieving the most stringent objectives, by performing stringent system engineering activities to produce expensive development artefacts, thus, increasing the development and certification costs. Allocating less stringent SILs to less-critical SPL components and most stringent SILs only to highly critical components can reduce both system's (variant) and SPL development and certification costs. A heuristic that expresses the relative cost jumps of developing critical system components to address different SILs, eg, in terms of DALs, can be used to support engineers in estimating system/SPL development costs according to the SILs allocated to system hazards and component failures.

For example, we can adopt the following cost heuristic to estimate the costs of developing the TFC-SPL: 0 to address DAL E, 10 for DAL D, 20 for DAL C, 40 for DAL B, and 50 for DAL A. We use this cost heuristic for illustrative purposes, but any other that engineers find more suitable can be used instead. By considering this cost heuristic, and where DAL A and DAL C are assigned to ensure the safe use of the PWM Decoder and the Barometric Processor subsystem components across the TFC-SPL (see column DAL-SPL from Table 2), the estimated cost for developing these TFC-SPL components is 70 (20 [DAL C] + 50 [DAL A]), instead of 100 in case of assignment of DAL A to both components. Thus, instead of allocating the most stringent DAL A assigned to a component failure to ensure the dependability of all components in a given usage context, eg, across the SPL, the decomposition of safety requirements may support engineers on structuring cost-effective Tiriba system and software product line development processes.

The SIL decomposition process allows engineers to assign objectives to a given component or a subset of components according to their level of integrity, instead of allocating highly stringent objectives to all subsystems/components of a system variant. Thus, the assignment of less stringent safety objectives to less critical components means less effort and costs need be considered for developing system components, addressing the safety requirements without incurring unnecessary costs and without compromising safety.

By analyzing the DALs allocated to mitigate the failure effects of components in the TFC-MAT system variant (see Table 2), we noted that the PWM Decoder is a highly critical component with DAL A. Thus, expensive objectives with independence, eg, SAE ARP 4754A - aircraft functional hazard assessment, and DO-178C - verification of additional code, should be addressed. Less critical components, eg, the Barometric Processor DAL C subsystem, demand less stringent objectives that do not require independence (see Table 2). Thus, the definition of development processes according to the level of integrity of each system component contributes to avoiding unnecessary costs for achieving compliance with standards without compromising safety.

Finally, it is important to highlight that when developing reusable system/software components, all variability aspects of each component should be considered from the initial stages of the SPL life cycle, and the most stringent SIL assigned to a given component in different usage contexts should be assigned/allocated to that component to ensure its safe use across the SPL, ie, across a range of system variants/configurations. Thus, product/system and contextual variability will not change the mitigation mechanisms for components in specific system variants.

### 5.2.3 Component fault modeling and propagation and product line variation

In the safety engineering life cycle, component fault analysis and modeling is intended to identify how software components contribute to the occurrence of potential system-level failures identified during hazard analysis and estimating the reliability, eg, in terms of failure and repair rates of hardware components. Variation in the system hazards and their causes can be further propagated to how components contribute to the occurrence of system hazards.

Variability in component fault modeling can be found in component output deviations that may contribute, in some way, to the occurrence of system-level failures, which might change from one targeted system and usage context variant to another (see column Output Deviation from
Table 3). Such variation can be further propagated to the causes of deviations in component outputs, which may change according the chosen system variant and usage context. Thus, different deviations in component inputs, component internal failures, or combinations of component input deviations and internal failure modes specified in a failure expression, which contribute to the occurrence of a given output deviation, may also be raised according to the targeted system and usage context variants (see column Failure Expression from Table 3).

Variation in the product line design (see Section 5.1), i.e., in architectural and behavioral modeling, may also impact on how component failures or faults propagate to other components. Output deviations of a given component may be propagated to different components according to the chosen system and usage context variants. In addition, variation in the design and usage context may also impact on different input deviations and internal failures that may contribute to the occurrence of the output deviations of a component, changing the way failures propagate inside the component (i.e., internal fault propagation).

Different system and usage context variants may lead to different connections among components, via component ports, which may change the way in which failures propagate both inside the component, and throughout the system architecture, i.e., external fault propagation (see Component Failure Data columns from Table 3). It is important to highlight that variation in the causes of output deviations (see column Failure Expression from Table 3) may impact on both internal and external failure propagation of the component, thus, changing how input and internal failures propagate inside the component, and how component output failures are propagated to other components of the system architecture. Finally, we can also find variability in the assignment of values for reliability attributes, e.g., failure and repair rates, to a given hardware component, which may change according to the targeted system variant.

Variability in hazard analysis and component fault modeling propagates to fail tree gates and nodes, and failure cut sets in FTA (see Figure 6), and in the way that components contribute, i.e., with direct or indirect failure modes, to the occurrence of the hazards in different system variants in FMEA (see Section 4.3.3). The analysis of FTAs and FMEA results for different system variants (see Section 4.3.3) provides the evidence of traceability of variation in the design and usage context throughout the cause-effect relationships between component-level failures and system-level failures. Such analysis provides insights into how SPL components can fail and contribute to the occurrence of hazards across different system variants, providing feedback for product line engineers addressing dependability, in terms of the safety and reliability, of the components across the SPL, i.e., across a range of system variants.

Finally, variation in hazard analysis, safety requirements, component fault modeling, FTA, and FMEA further propagate to the structure of product assurance cases. We can automatically generate Assurance Cases, with the support of model-based techniques, for a specific system variant from design and dependability assets. An Assurance Case is a defensible, comprehensive, and justifiable argument supported by a body evidence, which demonstrate that the system is acceptably safe to operate in a particular context. Automotive and aerospace standards and considerations and regulations recommend the production of an Assurance Case for certification of critical systems. The automatic generation and the analysis of the impact of variability on the structure of product assurance cases are outside the scope of this paper.

6 RELATED WORK

Research on variability management with dependability artefacts is split into extensions of traditional safety and dependability analysis techniques, e.g., Fault Tree Analysis and Failure Modes and Effects Analysis, to suit Software Product Line Engineering processes, and model-based techniques to support dependability analysis integrated with the system design. The most notable work in the first category is the extension of Software Fault Tree Analysis (SFTA), named Product Line SFTA, to address the impact of SPL variation on system dependability analysis. In the Product Line SFTA approach, each leaf node of a software fault tree is labeled with commonality or variability associated with that leaf node. The Product Line SFTA approach is built upon a technique for developing a product line SFTA in the domain engineering phase and a pruning technique for reusing the SFTA in the application engineering phase.

The Product Line SFTA approach is further extended to integrate SFTA results with state-based models. This extension allows the mapping of software fault tree leaf nodes into components and modeling the behavior of each component in a state chart. The Product Line SFTA and its state-based modeling extension considers Fault Tree Analysis as a reusable asset. However, fault trees can be automatically generated from dependability information produced during hazard analysis and component fault modeling in the domain engineering phase. Thus, earlier variability management of dependability properties on FTA and FMEA synthesis, as we presented in this paper, enables the systematic reuse of dependability information and traceability of dependable-related variation throughout the SPL safety life cycle. Fault trees and FMEA artefacts can be automatically generated for a particular system variant, with the support of compositional dependability analysis techniques, from the reused dependability information produced in the DEpendable-SPLE application engineering phase.

In the second category, Schulze et al. have proposed an approach that integrates commercial Medini ISO 26262 compliant safety analysis and pure::variants tools, to enable support for variability management in functional safety-related assets, evaluated in an automotive case study. The Schulze et al approach is based on a referencing model, which maps problem-domain features with artefacts in the solution space, in this case, requirements, fault trees, and safety goals. The Schulze et al approach was further extended with a process to support model-based change impact analysis of variability in automotive functional safety. This process combines variability management techniques with safety engineering and software configuration management activities to achieve a complete safety assessment. The proposed process supports change impact analysis in
the following scenarios: (a) when a specific variant shows undesired behavior and it needs to be fixed, (b) in cases where an innovative function requires an extension of an existing system function, and (c) when the function behavior is changed, and it should be analyzed, or when a newer optional function is developed.

Similar to the approach of Schulze et al and its extension, the approach proposed by the authors in this paper is built upon an SPL variability model, linking problem-domain context and system features with artefacts from the solution space, eg, components and their failure data. Although the Schulze et al\textsuperscript{11} approach provides support for variability management in functional safety assets, they did not emphasize the management of the impact of contextual variation on architectural and behavioral models, HARA and component fault modeling, as we presented in this paper. In addition, the author's approach is applicable to domains other than automotive, eg, aerospace, and it is independent, from underlying dependability analysis and variability management tooling support. Nevertheless, the Schulze et al approach\textsuperscript{11} and its extension\textsuperscript{51} also provides a good and efficient solution for variability and change management on functional safety.

Kaßmeyer et al\textsuperscript{52} proposed a systematic model-based approach integrated with a change impact analysis approach.\textsuperscript{51} The Kaßmeyer et al tool-supported approach combines requirements engineering and architectural design, safety analysis, and variability management tools, allowing seamless safety engineering across product variants by representing safety artefacts in a homogeneous UML-compliant modeling notation. In their approach, HARA and component fault modeling is performed by annotating the UML model in the same way as in the DEPendable-SPLE approach.

As part of Kaßmeyer et al approach,\textsuperscript{52} Domis et al\textsuperscript{10} have extended the Component Integrated Component Fault Trees (C2FT) technique with variation points and integrated it with UML via a profile in the Enterprise Architect tool. Although the Kaßmeyer et al approach\textsuperscript{52} and its extension\textsuperscript{10} also provides a good solution for variability management in functional safety, they are dependent upon specific commercial tools. On the other hand, DEPendable-SPLE approach can be applied independently from underlying variability management and dependability analysis tooling support, ie, engineers can apply the approach steps in their projects with the support of different compositional modeling and variant management techniques, eg, Simulink/HiP-HOPS and pure::variants,\textsuperscript{11} other than those used by the authors in the case study.

7 | CONCLUSION

In this paper, we presented the DEPendable-SPLE approach that extends conventional SPL processes with support for variability management in both design and dependability analysis. The DEPendable-SPLE approach enables the systematic reuse of both SPL architectural, behavioral, and dependability models in the application engineering phase. DEPendable-SPLE is also applicable independently from underlying variability management and dependability analysis techniques and tools. In this paper, we performed DEPendable-SPLE approach steps with the support of OSATE AADL and Error Annex system modeling and compositional dependability analysis toolset, BVR variability management toolset, and OSATE AADL adapter for BVR toolset developed by the authors.

The AADL adapter to the BVR tool was developed to enable BVR to communicate with OSATE model editors for managing variability in AADL design and dependability/error models. We used these tools to support both dependability analysis and variability management steps in the Tiriba flight control SPL. DEPendable-SPLE supports the analysis of the impact of design and usage context variations on dependability analysis artefacts. Thus, we linked the Tiriba Flight Control SPL system and usage context variants to their realization in both architecture, behavioral, and dependability models. Further, we generated, with the support of the BVR toolset, multiple variant-specific design and dependability models, during the product derivation process. Thus, we achieved the systematic reuse of architectural, behavioral, and dependability models early in the SPL safety process.

DEPendable-SPLE enabled the systematic reuse of almost 80% of TFC dependability information, produced in the domain engineering phase, in the derivation of each one of the four TFC system variants. This reduces the effort and costs for performing dependability analysis activities for a specific system variant. With the support of compositional techniques, in this case, OSATE AADL and Error Annex,\textsuperscript{7} fault trees, and FMEA were generated from the reused AADL design and AADL Error Annex dependability/error models. We also presented a detailed analysis of the impact of design and usage context variations on both design and dependability analysis (ie, hazard analysis and safety requirements and component fault propagation). Such analysis enables the systematic reuse of dependability assets produced during the domain engineering phase. Additionally, we presented a preliminary analysis of the impact of variation on allocation of safety requirements, expressed in terms of safety integrity levels, and their decomposition throughout components and their failure modes that may support engineers on structuring cost-effective development processes for both SPL and/or specific system variants.

In further work on this topic, we intend to focus on detailing how variability in AADL design and error models is specified and managed, and performing a user study of the BVR tool and AADL adapter developed by the authors and a comparison with existing variant management techniques. We also intend to perform a deeper investigation about the impact of design and usage context variations on SIL decomposition and development processes, and time-dependent, failure and repair behaviors of reconfigurable critical systems and software product lines. We intend to investigate the implications of SPL and system version/configuration variability on both safety and security analysis. Finally, we intend to investigate the use of model-driven techniques to generate variant-specific assurance cases, and the potential of SIL decomposition techniques in supporting engineers to take architectural decisions in the design of reconfigurable systems and safety-critical SPLs.
ACKNOWLEDGMENTS

This work was partially funded by CNPq Grant No. 152693-2011-4 and CAPES Brazilian research agencies and by the Post-doctoral program in Computer Science from the University of Sao Paulo Grant No. 2016.1.1027.55.4. The authors would like to thank AGX Technology for their support in the case study and the reviewers for their valuable comments that helped to improve the clarity and completeness of the paper.

ORCID

André Luiz de Oliveira https://orcid.org/0000-0003-0564-0034
Rosana Braga https://orcid.org/0000-0003-4259-0785
Paulo Masiero https://orcid.org/0000-0002-2541-4045
David Parker https://orcid.org/0000-0003-1482-1355
Yiannis Papadopoulos https://orcid.org/0000-0001-7007-5153
Ibrahim Habli https://orcid.org/0000-0003-2736-8238
Tim Kelly https://orcid.org/0000-0002-7385-2031

REFERENCES

1. Avizienis A, Laprie JC, Randell B. Fundamental Concepts Of Dependability. Technical Report 010028, University of California, USA; 2001.
2. ISO. ISO 26262: Road Vehicles Functional Safety; 2011.
3. RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification; 2012.
4. EUROCAE. ARP4754A - guidelines for Development Of Civil Aircraft and Systems; 2010.
5. Habli I, Kelly T, Hopkins I. Challenges of establishing a software product line for an aerospace engine monitoring system. In: Proceedings of the 11th International Software Product Line Conference; 2007: Japan:193-202.
6. Oliveira AL, Braga R, Masiero PC, Papadopoulos Y, Habli I, Kelly T. Model-based safety analysis of software product lines. International Journal of Embedded Systems. 2016;8(5/6):412-426.
7. Delange J, Feiler P. Architecture fault modeling with the AADL error-model annex. In: Proceedings of the 40th IEEE EUROMICRO; 2014; Verona, Italy:361-368.
8. Papadopoulos Y, Walker M, Parker D, Rüde E, Hamann R, Uhlig A, Gratz U, Lien R. Engineering failure analysis and design optimization with hip-hops. Journal of Engineering Failure Analysis. Elsevier. 2011;18(2):590-608.
9. Mazzini S, Favaro J, Puri S, Baracchi L. Chess: an open source methodology and toolset for the development of critical systems. In: Joint Proceedings of the EduSymp; 2016:59-66.
10. Domis D, Adler R, Becker M. Integrating variability and safety analysis models using commercial uml-based tools. In: Proceedings of the 19th International Software Product Line Conference, ACM; 2015; Nashville, USA:225-234.
11. Schulze M, Mauersberger J, Beuche D. Functional safety and variability: can it be brought together? In: Proceedings of the 17th International Software Product Line Conference. New York, USA; ACM; 2013:236-243.
12. Dordowsky F, Bridges R, Tschope H. Implementing a software product line for a complex avionics system. In: Proceedings of the 15th International Software Product Line Conference. Munich, Germany: IEEE; 2011:241-250.
13. Dehlinger J, Lutz R. Software fault tree analysis for product lines. In: Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering; 2004; Tampa, USA:12:21.
14. Feng Q, Lutz R. Bi-directional safety analysis of product lines. Journal of Systems and Software. 2005;78(2):111-117.
15. Gomez C, Peter L, Sutor A. Variability management of safety and reliability models: an intermediate model towards systematic reuse of component fault trees. In: Proceedings of the 29th Int. Conference Computer Safety, Reliability and Security; 2010; Vienna, Austria:28-40.
16. Liu J, Dehlinger J, Lutz R. Safety analysis of software product lines using stated modeling. J. of Systems and Software. 2007;80(11):1879-1892.
17. Aeronautics National, (NASA) SpaceAdministration. Fault Tree Analysis with Aerospace Applications. Technical report, an update to NUREG-0492, NASA Office of Safety and Mission Assurance, Washington, DC, USA; August, 2002.
18. Kang KC, 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 Engineering. 1998;5(1):143-168.
19. Krueger C. Variation management for product line development. In: Proceedings of the 16th International Software Product Line Conference; 2007:37-48.
20. Gomaa H. Designing Software Product Lines with uml: from Use Cases to Pattern-Based Software Architectures. Redwood City, CA, USA: Addison Wesley Longman Publishing; 2004. ISBN 0201775956.
21. Leveson NG. Safeware: System Safety and Computers, Addison-Wesley, Reading, MA; 1995.
22. Oliveira AL, Braga RTB, Masiero PC, Papadopoulos Y, Habli I, Kelly T. Variability management in safety-critical software product line engineering. In: Proceedings of the 17th International Conference on Software Reuse, New Opportunities for Software Reuse. Lecture Notes in Computer Science, vol 10826; 2018; Madrid, Spain:3-22.
23. MoD. DEF-STAN 00-56 Issue 4 Part 1: Safety Management Requirements for Defense Systems. Technical report, UK Ministry of Defense; 2007.
24. Braga RTV, Trinidad Jr. O, Branco KRLJC, Lee J. Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In: Proceedings of the 16th International Software Product Line Conference; 2012; Salvador, Brazil:1-10.
25. Branco K, Pelizzoni J, Trinidad O, Osorio F, Wolf D. Tiriba: a new approach of UAV based on model-driven development and multiprocessors. In: Proceedings of the International Conference on Robotics and Automation (ICRA). Shanghai, China: IEEE; 2011:1-4.
26. Lee K, Kang KC. Usage context as key driver for feature selection. In: Proceedings of the 14th Int. Software Product Line Conf.; 2010:32-46.
27. ICAO. Circular 328 AN/190: Unmanned Aircraft Systems Operations (UAS). International Civil Aviation Organization (ICAO), Montreal, Canada; 2011.

28. Hartmann H, Trew T. Using feature diagrams with context variability to model multiple product lines for software supply chains. In: Proceedings of the 12th International Software Product Line Conference; 2008; Limerick, Ireland:12-21.

29. Mens K, Capilla R, Cardozo N, Dumas B. A taxonomy of context-aware software variability approaches. In: Companion Proceedings of the 15th International Conference on Modularity (MODULARITY Companion 2016); 2008; Malaga, Spain:119-124.

30. Capilla R, Ortiz O, Hinchee M. Context variability for context-aware systems. IEEE Computer. 2014;47(2):85-87.

31. Pohl K, Bockle G, Van Der Linden FJ. Software Product line Engineering: Foundations, Principles and Techniques. Secaucus, NJ, USA: Springer-Verlag New York; 2005.

32. BigLever. Gears. http://www.biglever.com. Accessed on: August 5th, 2018.; 2018.

33. Haugen O, Moller-Pedersen B, Oldevik J, Olsen GK. Adding standardized variability to domain specific languages. In: Proceedings of the 11th International Software Product Line Conference; 2008; Limerick, Ireland:139-148.

34. Heuer A, Pohl K. Structuring variability in the context of embedded systems during software engineering. In: Proceedings of the 8th Workshop on Variability Modeling of Software-intensive Systems, ACM; 2014.

35. Steiner EM, Masiero PC, Bonifacio R. Managing SPL variabilities in UAV simulink models with pure::variants and hephaestus. CLEI Electronic Journal. 2013;16(1):1-16.

36. Vasilevskiy A, Haugen O, Chauvel F, Johansen MF, Shimbara D. The BVR tool bundle to support product line engineering. In: Proceedings of the 19th International Software Product Line Conference, ACM; 2015; New York, USA:380-384.

37. Oliveira AL, Braga RTB, Masiero PC, Papadopoulos Y, Habli I, Kelly T. A model-based approach to support the automatic safety analysis of multiple product line products. In: Proceedings of the 4th Brazilian Symposium on Computing Systems Engineering, IEEE; 2014;7-12.

38. Administration FederalAviation. MIL-STD-882E - Department of Defense Standard Practice for System Safety, FAA; 2012.

39. Habli I. Safety standards: Chronic challenges and emerging principles. Handbook of Safety Principles. 2017:732-746.

40. Knight J. Safety Standards: A New Approach, Safety-Critical Systems Club; 2014.

41. Bieber P, Delmas R, Seguin C. Dalcus theory and tool for development assurance level allocation. In: Proceedings of the 30th International Conference on Computer Safety, Reliability and Security; September, 2011; Naples, Italy:43-56.

42. Braga R, Branco KRLJC, Trindade Jr. O, Masiero PC. The proliCES approach to develop product lines for safety-critical embedded system and its application to the unmanned aerial vehicles domain. CLEI Electronic Journal. September, 2012;15(2):1-12.

43. Braga RTV, Trindade Jr. O, Branco KRLJC, Neris LO, Lee J. Adapting a software product line engineering process for certifying safety critical embedded systems. In: Proceedings of the 31st International Conference on Computer Safety, Reliability and Security; September, 2012; Magdeburg, Germany:352-363.

44. MATHWORKS. MATLAB Simulink. http://www.mathworks.com/products/simulink. Accessed on: August 10th, 2018; 2018.

45. ECLIPSE. Eclipse Modeling Framework Project. http://www.eclipse.org/modeling/emf. Accessed on: August 5th, 2018; 2018.

46. Oliveira AL. Tiriba Product Line Dependability Analysis. Available on-line: https://github.com/aloliveira/tiriba/blob/master/CaseStudyTiriba.pdf. Accessed on: February 18th, 2019; 2019.

47. EUROCAE. Aircraft Wheel Braking System. https://github.com/osate/examples/tree/master/ARP4761. Accessed on: August 5th, 2018; 2018.

48. Leveson. Door Control System. https://github.com/osate/examples/tree/master/Train. Accessed on: August 5th, 2018; 2018.

49. Hawkins R, Habli I, Kolovos D, Paige R, Kelly T. Weaving an assurance case from design: a model-based approach. In: Proceedings of the 16th High Assurance Systems Engineering, IEEE; 2015; Daytona Beach, USA:110-117.

50. Kelly T, McDermid J. Safety case construction and reuse using patterns. In: Proceedings of the 16th International Conference on Computer, Conference Computer Safety, Reliability and Security, Springer; 1997:55-69.

51. Kaßmeyer M, Schulze M, Schurius M. A process to support a systematic change impact analysis of variability and safety in automotive functions. In: Proceedings of the 19th International Software Product Line Conference. Nashville, USA: ACM; 2015:235-244.

52. Kaßmeyer M, Moncada DSV, Schurius M. Evaluation of a systematic approach in variant management for safety-critical systems development. In: Proceedings of the 13th International Conference on Embedded and Ubiquitous Computing, IEEE; 2015:35-43.

How to cite this article: Oliveira AL, Braga R, Masiero P, et al. Variability management in safety-critical systems design and dependability analysis. J Softw Evol Proc. 2019;31:e2202. https://doi.org/10.1002/smr.2202