An ontology-based approach to support the certification of Safety-Critical Software Product Lines

Lucas Bressan[1], Regina Braga[1], Fernanda Campos[1], André L. de Oliveira[1]

[1]Programa de Pós Graduação em Ciência da Computação– Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brazil

`lucasbressan@ice.ufjf.br, {regina.braga, fernanda.campos}@ufjf.edu.br, andre.oliveira@ice.ufjf.br`

Abstract. *Safety-critical Software Product Lines (SPLs) are required to demonstrate compliance with domain-specific safety standards. Different component configurations may require the inclusion or exclusion of certain features depending on their impact on safety. Additionally, variants may present distinct criticality levels which imply in different safety requirements during their development and evaluation. Some authors have proposed approaches to address safety certification taking into account SPL Engineering (SPLE) activities. Those can be however, labor intensive and impracticable when dealing with larger and complex product lines. In this paper, we propose an ontology-based approach to support safety engineers on identifying features and assets relevant for the configuration and certification of Safety-Critical Product Lines. The approach was evaluated, considering a realistic SPL from the aerospace domain and the DO-178C safety standard. As a result, the application of the proposed approach was proven to support the traceability of SPL requirements and certification levels, thus, reducing the complexity of the deployment of different component configurations.*

1. Introduction

Critical systems are computer systems in which failures may lead to catastrophic consequences. Due to their critical nature, the development of these systems often demand compliance with domain-specific safety standards. Standards, such as the DO-178C [RTCA 2011] for avionics and ISO26262 [ISO 2018] for automotive systems, establish a set of safety requirements that provide evidence that a piece of software is acceptably safe. Safety standards establish a set of safety integrity levels to classify the criticality of individual systems and components. Integrity levels are assigned based on the results obtained through system safety engineering and analysis activities.

Due to the benefits of large-scale reuse, increased quality and reduced development costs introduced by Software Product Lines (SPLs), Software Product Line Engineering (SPLE) [Pohl et al. 2005] has been widely adopted in many different critical industry domains such as automotive, aerospace and railway. Traditional SPLE however, does not support safety engineering activities. Therefore, safety engineering, analysis and certification activities must be integrated into SPLE when considering safety-critical SPLs [Habli and Kelly 2007][Kelly et al. 2016].

Safety standards such as the DO-178C only consider the certification of individual products thus, being limited when it comes to the certification of variant-intensive and families of critical systems. Due to that limitation, authors such as Braga et. al. [Braga et al. 2012] have proposed ways to incorporate certification aspects into SPL Engineering. Despite proven to be effective, existing SPL engineering-based approaches for critical systems lack on providing more efficient strategies to manage and enable the traceability required to keep track and manage aspects related to system design and safety. Such traceability might be crucial when dealing with bigger and complex reconfigurable software platforms.

In this paper, we present an ontology-based approach to support identification of features and safety requirements relevant to the configuration and safety certification of Safety-Critical Software Product Lines. The main goal of the approach is to reduce the effort related to the deployment of different Safety-Critical software component configurations while considering safety and certification. The proposed approach was evaluated through a feasibility study using the Tiriba UAV SPL, a realistic Safety-Critical SPL from the aerospace domain and guidance provided in the DO-178C safety standard. The results pointed to the feasibility of the proposed approach.

This paper is organized as follows: **Section 2** contains the background. **Section 3** discusses the related works. **Section 4** provides an overview of the proposed ontology-based approach. **Section 5** illustrates the evaluation of the proposed approach in a realistic safety-critical SPL of the aerospace domain. Finally, **Section 6** presents the conclusions and future work.

2. Background

Critical systems are computer systems in which, failures may result in catastrophic consequences. Due to benefits delivered by Software Product Lines [Pohl et al. 2005], such as enhanced product quality, large-scale reuse and shorter time-to-market, Software Product Line (SPL) Engineering is being increasingly adopted in the industry [Villela et al. 2014].

SPL Engineering is split into two phases: Domain and Application Engineering. The Domain Engineering phase enables the establishment of the reusable platform by supporting the definition of commonalities and variability within the product line. The reusable platform includes artifacts such as documentation, requirements, design, realization, source code, test cases and the feature model. Feature models are used to describe systems in terms of features and to specify their points of variability. Features are distinct system characteristics visible to the end user [Lee et al. 2002]. The Application Engineering phase supports the configuration and derivation of different products based on the information within the reusable platform. It ensures the appropriate binding of variability, based on the product requirements [Pohl et al. 2005].

Even though the adoption of SPL Engineering in the development of product families has its many advantages, additional practices must be taken into account when dealing with safety-critical Product Lines [Habli and Kelly 2007]. In safety-critical systems, different product configurations must address different certification requirements depending on their criticality. Therefore, the integration of safety engineering and certification tasks into SPL Engineering is considered crucial [Braga et al. 2012][Oliveira et al. 2018]. DEPendableSPLE extends the conventional SPLE methods, to support safety engineering activities on both the Domain and Application Engineering phases of SPL Engineering. It provides guidance for the definition of the safety-critical reusable platform, product configuration and product-specific safety analysis. It not only enables the specification and binding of variability at design level but also, at safety engineering and assessment level, covering safety engineering phases such as Hazard Analysis and Risk Assessment (HARA). A hazard is a potential source of harm caused by the malfunctioning behavior of a system, components and its functions [ISO 2018]. HARA consists of identifying the potential hazards that may affect the behavior of a system or component, estimating their risk and criticality. Such criticality can be measured in terms of quality attributes e.g.: severity, controllability and quantitative metrics e.g.: availability and reliability.

Safety standards provide guidance to ensure the safety of a system and its compliance with a targeted criticality level. The DO-178C [RTCA 2011] lists a series of requirements to guide the development and assessment of airborne software systems. It contains a total of five different criticality levels named Development Assurance Levels (DALs) or Software Levels. The rules and recommendations concerning DAL / Software Levels allocation are presented in a complementary document, the SAE ARP 4754A [Aerospace 2010].

Criticality levels range from A to E and are allocated to software components, according to the risks and severity associated to their hazards. The risk of a hazard is calculated based on the number of times it happens per flight hour. Severity is purely qualitative and can be set as minor, major, hazardous or catastrophic. Minor hazards describe failures that may cause a routine flight plan change. Major hazards describe failures that may lead to passenger discomfort or significant increases in crew workload. Hazardous failures have a large negative impact on safety by reducing the ability to operate the aircraft correctly and may cause serious or fatal injuries to passengers and crew. Catastrophic failures are those that may lead to a crash e.g.: loss of control or function to safely operate the aircraft. Certification requirements change according to the different criticality levels. Therefore, higher Software Levels demand a greater number of development and assessment activities thus, increasing development costs and effort.

3. Related Works

Braga et al. [Braga et al. 2012] describe how certification may impact the feature modeling and safety requirement allocation in Safety-Critical Product Lines from the Aerospace Domain. The main idea is to identify features that impact the certification of component variants while considering DO-178B Software Levels. The approach however, does not consider multi-level traceability between different configurations and features. When specifying the feature model, the authors have only considered the direct relationships between features and, although it is still possible to manually establish traces between them, doing so, can prove to be time consuming and labor intensive when dealing with bigger and more complex systems.

Even though the adoption of a development focused on Software Product Lines (SPL), large-scale reuse and families of systems may present many advantages, adaptations to the Software Product Line Engineering (SPLE) paradigm are necessary to include safety engineering and assessment tasks into the development life-cycle.

Variations in design and context can impact the artifacts generated through safety engineering phases such as Hazard Analysis and Risk Assessment (HARA). Moreover, different HARA results may also impact the allocation of safety requirements. In order to support the SPL Engineering-based development in critical domains, Oliveira et al. [Oliveira et al. 2018] present DEPendableSPLE, an adaptation of the traditional SPL Engineering methods to include variability management, reuse and traceability between artifacts generated through safety engineering tasks and product design.

Filho et al. [Filho et al. 2012], introduce new ways to semantically enrich feature models, using ontologies. The approach considers aspects such as multiplicities, optionality and the establishment of relationships between features and elements such as requirements, code and test cases. Despite doing so, the approach has not been applied or extended yet, to consider artifacts and relationships relevant to the development and assessment of safety-critical SPLs e.g.: safety engineering, analysis results and the allocation of safety requirements.

4. The Ontology-based Approach

This section presents the main aspects regarding the proposed approach. The approach comprises the following phases: Reusable Platform Specification, Inference Generation and Component Configuration. The output of these phases are used and processed by the ontology to generate the desired inferences. The approach is depicted in **Figure 1** and described in the following.

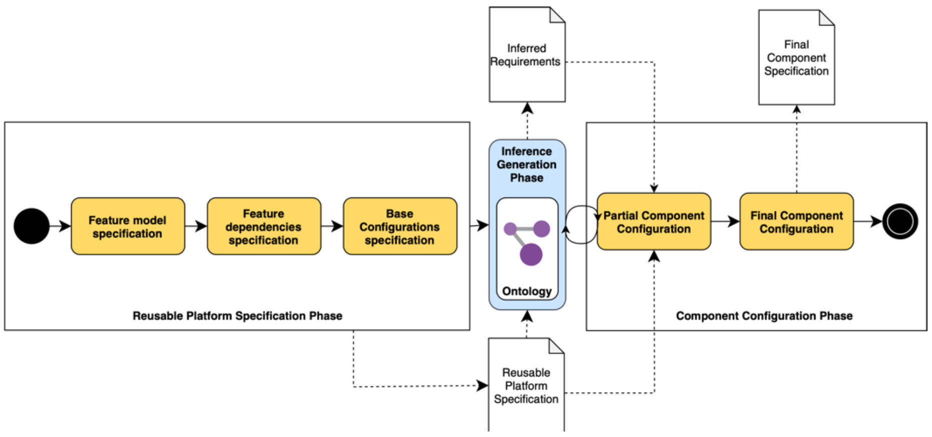

Inferred Requirements
Final Component Specification
Inference Generation Phase
Feature model specification
Feature dependencies specification
Base Configurations specification
Ontology
Partial Component Configuration
Final Component Configuration
Reusable Platform Specification Phase
Component Configuration Phase
Reusable Platform Specification

Figure 1 - The proposed ontology-based approach

4.1. Reusable Platform Specification Phase

The Reusable Platform Specification phase covers the application of Domain Engineering activities to specify the reusable platform of the desired variant-intensive software component. In this phase, engineers determine how certain software functionalities may be combined into features and how these features relate to each other through dependency relationships e.g.: "**Feature1** *requires* **Feature2**". Moreover, a preliminary set of dependencies between certification-relevant features, base configurations and integrity levels are defined.

In the proposed approach, the certification-relevant features are those required, excluded or optional by a base configuration, so it achieves a certain integrity level. Such relationships are mainly obtained through Hazard Analysis and Risk Assessment

(HARA) techniques such as Reliability estimation, Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA). If the inclusion of a feature into the configuration does not affect the desired Software Level, then such feature can be considered optional. If the inclusion of such feature increases the risks associated with the considered component and can possibly increase desired Software Level, then this feature must be excluded from the desired configuration or classified as *worksAgainstCert*. At last, features that when added reduces the risks associated with a component or help to ensure the desired Software Level can be set as *requiresForCert* or *worksForCert*.

Base configurations comprise a set of features that can partially describe a software component variant configuration. These configurations are always detailed in terms of their base features e.g.: **BaseConf1** *implements* {**F1** *and* **F2** *and* **F3**} and possible criticality levels e.g.: **BaseConf1** *hasCriticality* **{A, B}**. They can be instantiated and further extended in the Component Configuration Phase, to implement optional features or include additional certification-relevant features depending on the desired Software Level and the feature suggestions provided during the Inference Generation Phase.

4.2. Inference Generation Phase

The Inference Generation Phase serves as a bridging point between the Reusable Platform Specification and the Component Configuration phases. It contains the ontology and is responsible for processing the provided information and deriving data relevant for the configuration and certification of different software component configurations.

Competency Questions (CQs) are derived in the earlier stages of ontology development and are questions which the ontology should be able to answer [Fernandes et al. 2011; Noy and Hafner 1997]. A set of CQs were defined to help us with the formalization of the ontology[1]: **CQ1:** Which features are relevant for the certification process, considering different product configurations and certification levels? **CQ2:** Which process objectives are required by each criticality? **CQ3:** Which process objectives are required by each system variant, considering their expected software certification level? **CQ4:** What are the implicit dependencies between features? **CQ5:** What are the implicit dependencies between product variants and features?

In addition to the competency questions, we have also considered the elements and feature relationships provided by [Braga et al. 2012] in their experience report when specifying the classes, object properties and relationship rules in the ontology. The proposed ontology was specified using the Protegè[2] tool and is depicted in **Figure 2**.

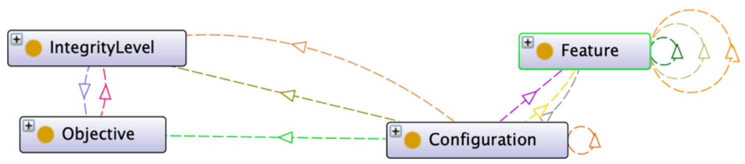


Figure 2 - The proposed ontology

[1] Available on: https://github.com/bressan3/Tiriba-UAV-SPL-Onthology
[2] https://protege.stanford.edu

Different component configurations, comprising a set of Feature instances, are represented by the Configuration class. The Objective and IntegrityLevel classes represent standard-specific aspects. Depending on the critical domain targeted by the SPL, different safety standards may be required in order to demonstrate safety compliance. Thus, Objective represents the objectives or safety requirements listed in the considered standard. Instances belonging to IntegrityLevel class, represent the documented criticality or integrity levels e.g.: Automotive Safety Integrity Levels (ASILs) in the ISO26262, Software Levels or DALs in the DO-178C.

Instances belonging to the Feature, may require, exclude or make other features optional. These relationships, are implemented through *requires*, *makesOptional* and *excludes* object properties. Configurations may implement a series of different features and therefore, relate to members of the Feature class through the implements property.

In addition to the classes and object properties described above, the ontology was also enriched with property chain and SWRL rules. If a feature **F1** *requires* **F2** *and* **F2** *requires* **F3**, then it is known that the selection of **F1** will automatically imply in the selection of both **F2** and **F3**. Thus, the *requires* object property characterizes the transitive property chain in order to address **CQ2**. Furthermore, if a component configuration **CompConf1** *requires* **F1** for achieving a certain criticality level and **F1** *excludes* **F2**, then **CompConf1** will also have to *exclude* **F2**. Thus, a SWRL rule to describe such relationship was also considered in the ontology so it can infer that rule and answer **CQ1**.

4.3. Component Configuration Phase

The component configuration phase implements activities related to the generation of different product variants according to information that has been previously provided in the Reusable Platform. The first step in this phase is the partial configuration of the desired component. As previously described in **Section 4.1**, different base component configurations must be specified in the reusable platform. This partial component configuration is fed into the ontology and new suggestions regarding the possible certification levels and the implicit feature requirements, are generated.

These suggestions will indicate which features are required, recommended or excluded by the desired base configuration and will be further used as input towards reaching a final software component configuration. With the suggestions generated by the ontology, project managers can include features, allocate different criticality levels and get real-time suggestions of the new safety requirements and component configuration possibilities upon doing so.

Features that help a base configuration reach a certain integrity level, can sometimes require or exclude other features. When adding them to the partial component configuration, the ontology will process the inserted data and return all the new configuration requirements and possibilities. As a result, product managers will be then able to evaluate the new features that must be added or excluded from the configuration against the project requirements. These results can be used to determine if it is actually worthwhile to add a recommended feature into the desired component configuration or not. Such decision can be made by considering a number of factors such as integration effort and costs.

The proposed ontology also provides information regarding the safety requirement compliance needs of component configurations. Project managers can use such information, on top of the feature requirements, to estimate the effort required for the certification of a component configuration and agree on project decisions.

5. Evaluation

In this section, we evaluate the application of the proposed approach in a real world scenario, through a viability study. The evaluation was conducted considering the description of the Tiriba UAV Software Product Line, its safety certification-related attributes provided by Braga et al. [Braga et al. 2012] and the DO-178C safety standard [RTCA 2011]. Furthermore we have also considered the guidance provided by Wohlin et. al. [Wohlin et al. 2012] in the presented study. The evaluation was divided into five steps: study definition, formalization, planning, execution and evidences presentation.

5.1. Study Definition and Formalization

In order to define the scope of this feasibility study, we must determine its purpose, point of view and the context. The Goal Question Metrics (GQM) approach proposed by Basili, Caldiera and Rombach [Basili et al. 1994] was used to define these attributes:

Analyze the ontology-based approach to support the configuration and certification of Safety-Critical Software Product Lines **with the purpose of** evaluating its feasibility on extracting implicit relationships that can support the configuration of different safety-critical software components **with respect to** safety certification **from the point of view of** software product lines **in the context of** safety-critical systems.

Based on the application of the GQM approach, the following research question was derived: **RQ:** How does the proposed approach help engineers on managing and configuring Safety-Critical Software Product Lines? Moreover, the following secondary research questions were derived: **RQ1:** How does the proposed approach help on the identification of implicit feature requirements when considering certification? **RQ2:** How does the proposed approach supports the identification of implicit feature dependencies upon the inclusion of new features into a component configuration? **RQ3:** How does the proposed approach support engineers on estimating the certification objective requirements when considering different component configurations?

5.2. Study Planning

An Unmanned Aerial Vehicle (UAV) is an aircraft which is not flown by an onboard human operator. UAVs comprise one of the main components within an Unmanned Aircraft System (UAS). Apart from including the UAV itself, UASs may also implement additional components such as a controlling station, payload and communication systems [ICAO 2011]. The Tiriba UAV Software Product Line and its feature model, have been modified by Braga et al. [Braga et al. 2012] from its original version, in a way which features were separated in different layers, according to their purpose based on Kang et al.'s [Kang et al. 1998] work.

The Usage Context Layer, which is the topmost abstraction in the Tiriba UAV feature model, comprises three different feature categories or variation points: *Application*, *UAVDimension* and *Airspace*. The *Application* variation point contains features that describe the domain which the UAV will be used in e.g.: Agriculture, Environment Monitoring and Defense. The features within *UAVDimension* are related to the size and the weight of the UAV component e.g.: Light, Small or Heavy. At last, the Airspace variation point contains features that describe the kind of airspace which the UAV will be certified to operate in e.g.: Controlled or Uncontrolled. Variants are specified considering the selection of one feature contained in each one of these three variation points. Accordingly, some possible configurations, when considering the Tiriba UAV software component are: *AgricultureSmallControlled*, *DefenseSmallControlled* and *DefenceLightUncontrolled*.

Figure 3 shows an excerpt of the Tiriba UAV feature model and its feature-to-feature relationships. Certain features may require, make optional or exclude others. The *Mission Abortion* feature for example, must be present whenever the *Controlled* airspace feature is selected. Therefore, variants such as *AgricultureSmallControlled*, must always implement the *MissionAbortion* feature. UAVs falling under the *Heavy* category, cannot be launched by *Hand* or implement the *Parachute Landing* functionality. *Parachute Landing* adds a new layer of redundancy in the landing procedure thus, making it safer. Therefore, UAV variants with lower Software Levels, may require or recommend the inclusion of such feature in their configurations.

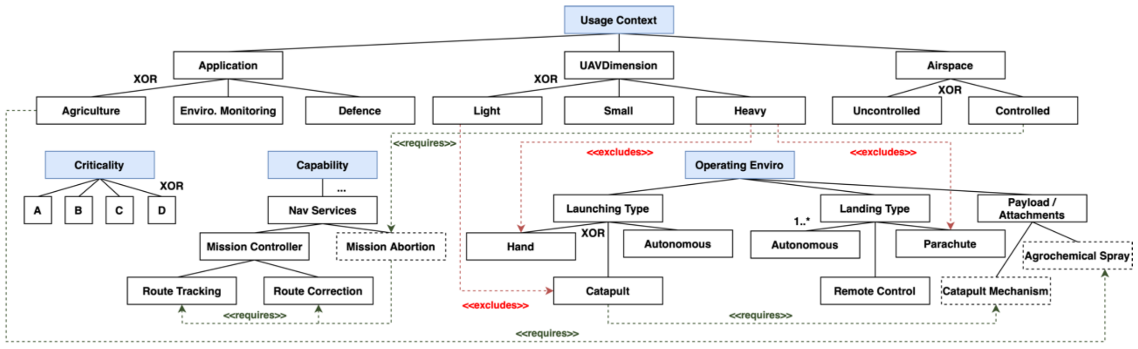


Figure 3 - An excerpt of the Tiriba UAV SPL feature model and its feature-to-feature relationships

5.3. Execution

In order to perform this feasibility study, the information regarding the Tiriba UAV Software Product Line such as its Features, Feature-to-Feature and Configuration-to-Feature Certification relationships were loaded into our reusable platform. The *AgricultureSmallUncontrolled* UAV component configuration with Software Level B was taken into account for this evaluation. The experiment was performed using Protegè and the Pellet reasoner. The ontology was populated with the information within the reusable platform, DO-178C software levels, certification phases, their objectives and UAV base configurations.

The *AgricultureSmallUncontrolled* UAV base component configuration implementing the Agriculture, Small and Uncontrolled context features, was fed into the ontology followed by the base configuration for the *AgricultureSmallUncontrolled*

UAV with expected Software Level B. **Table 1** shows the certification and the DO-178C software level dependencies required by the *AgricultureSmallUncontrolled* UAV configuration to achieve compliance with the DO-178C Software Level B. These required, desirable and excluded features were obtained through the results of safety analysis activities during Hazard Analysis and Risk Assessment (HARA):

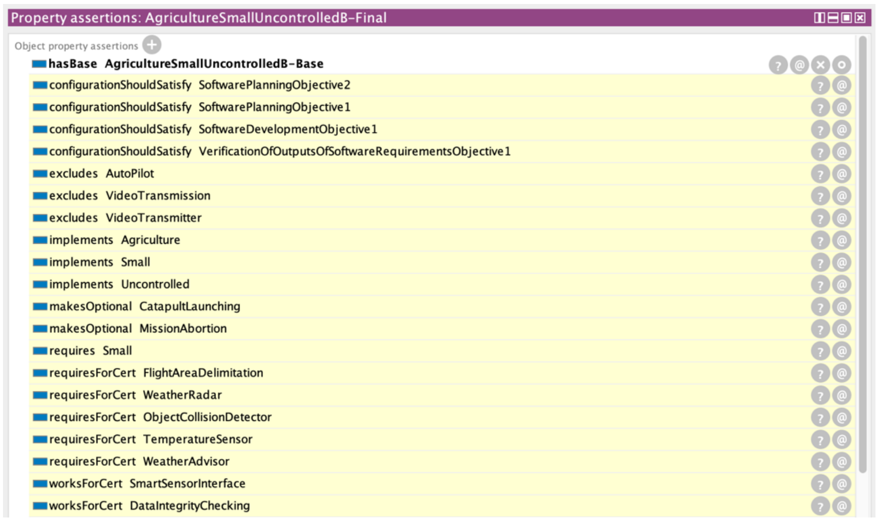
Table 1 - The list of certification-relevant features required by the AgricultureSmallUncontrolled UAV component configuration for achieving Software Level B

Relationship	Individual	Class
hasCriticality	B	Criticality
requiresForCert	SmartSensorInterface	Feature
requiresForCert	FlightAreaDelimitation	Feature
worksForCert	DataIntegrityChecking	Feature

A preliminary component configuration based off the *AgricultureSmallUncontrolled* UAV configuration with Software Level B was created. At last, the Pellet reasoner was executed on the populated ontology, considering its instances and relationships.

5.4. Evidences Presentation

Once having the inferences generated and back propagated into the ontology by the Pellet reasoner, the following feature suggestions and certification requirements were generated for the preliminary AgricultureSmallUncontrolled UAV component configuration, with Software Level B:


Property assertions: AgricultureSmallUncontrolledB-Final

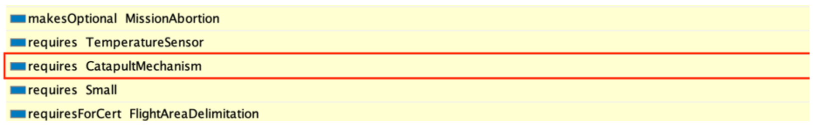
Object property assertions

```
hasBase  AgricultureSmallUncontrolledB-Base
configurationShouldSatisfy  SoftwarePlanningObjective2
configurationShouldSatisfy  SoftwarePlanningObjective1
configurationShouldSatisfy  SoftwareDevelopmentObjective1
configurationShouldSatisfy  VerificationOfOutputsOfSoftwareRequirementsObjective1
excludes  AutoPilot
excludes  VideoTransmission
excludes  VideoTransmitter
implements  Agriculture
implements  Small
implements  Uncontrolled
makesOptional  CatapultLaunching
makesOptional  MissionAbortion
requires  Small
requiresForCert  FlightAreaDelimitation
requiresForCert  WeatherRadar
requiresForCert  ObjectCollisionDetector
requiresForCert  TemperatureSensor
requiresForCert  WeatherAdvisor
worksForCert  SmartSensorInterface
worksForCert  DataIntegrityChecking
```

Figure 4 - The preliminary *AgricultureSmallUncontrolled* UAV configuration and its certification and feature requirements

The generated results illustrate not only the relationships that had been manually modeled in both the *AgricultureSmallUncontrolled* and the AgricultureSmallUncontrolled UAV with Software Level B base configurations but also, the certification objectives that the desired software component configuration should satisfy and its implicit feature requirements.

By analyzing **Figure 4**, we can observe that the CatapultLaunching and WeatherRadar features are optional in the configuration we have so far. Since these features may impact in additional feature requirements, new features can become mandatory, upon integrating them into the current configuration. **Figure 5** displays the new feature requirements brought up by integrating the CatapultLaunching and WeatherRadar features into the preliminary *AgricultureSmallUncontrolled* UAV Software Component configuration:



makesOptional MissionAbortion
requires TemperatureSensor
requires CatapultMechanism
requires Small
requiresForCert FlightAreaDelimitation

Figure 5 - The new implicit feature requirements upon adding new features to the preliminary *AgricultureSmallUncontrolled* UAV configuration

Considering the results obtained throughout the evaluation so far, we can now answer the research questions listed earlier in **Section 5.1**:

RQ1 - How does the proposed approach help on the identification of implicit feature requirements when considering certification? The approach support the identification of implicit certification feature requirements by displaying all the features that contribute, work against or are required so a configuration can achieve a certain software level. **Figure 4** lists a set of features and relationships which, although not being explicitly modeled into the base model, are implicitly required or desirable by the *AgricultureSmallUncontrolled* UAV so it can achieve Software Level B.

RQ2 - How does the proposed approach help on the identification of implicit feature dependencies upon the inclusion of new features into a component configuration? The approach also support the estimation of feature requirements that are not necessarily linked to certification. As previously shown in **Figure 5**, the inclusion of new features into the preliminary component configuration provides product managers with real-time information regarding the new implicit feature requirements that should also be considered alongside their decision.

RQ3 - How does the proposed approach help on estimating the certification objective requirements when considering different component configurations? The safety certification objectives can be gathered from as far back as when configuring the base model that will relate a software component configuration to an specific integrity level. As pictured in **Figures 4** and **5**, the ontology is able to provide, through its reasoning, the standard objectives that must be accomplished, in order to achieve the safety certification of the desired software component configuration.

RQ - How does the proposed approach help engineers on managing and configuring Safety-Critical Software Product Lines? At last, by considering the answers provided in **RQ1**, **RQ2** and **RQ3** and the collected evidence, we can conclude that the presented ontology-based approach, does indeed provide the intended support for the configuration, deployment and certification of different safety-critical software component configurations.

The first characteristic that sustains such affirmation has to do with the fact that the ontology, can provide project managers with relevant information regarding the different configuration possibilities and needs a software component can have. By doing so, it helps them decide if a certain feature should be added into a configuration or not. In the case of a feature that if included, would help a component achieve its desired integrity level, this decision can be made upon analyzing the impact that adding the suggested feature would have on certification and measuring if the benefits it introduces into the configuration, would justify the extra time and effort needed to integrate it. In addition, analysts can also use of the inferred information such as safety standard objectives and the certification feature requirements to better estimate the effort necessary to integrate, deploy and certify the desired component configuration.

6. Concluding Remarks and Future Work

This paper has presented an ontology-based approach to support the identification of features and safety requirements relevant to the configuration and safety certification of Safety-Critical Software Product Lines. We have considered a previous experience applied during the definition of a critical SPL of the aerospace domain, and presented an ontology-based approach to support project managers on configuring and deploying different software component configurations based on it.

The approach was evaluated through a feasibility study. A set of research questions were determined using the GQM method. The approach was then evaluated considering a realistic aerospace Safety-Critical Software Product Line and the DO-178C safety standard. As a result, we believe that, integrating the proposed approach into the reusable platform specification and resolution phases of Safety-Critical SPLs may reduce even more, the time and effort needed to analyze and make decisions when configuring, deploying and certifying different component or product configurations.

As future work, we will extend the proposed approach to support the automatic Hazard Analysis and Risk Assessment of base configurations through its integration with external safety analysis tools and a smarter and more automated estimation of the possible base configuration criticality levels and their specific certification relationships and requirements.

7. References

Aerospace, S. A. E. (2010). *SAE ARP 4754A: Guidelines for development of Civil Aircraft and Systems*. SAE International.

Basili, V. R., Caldiera, G. and Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of Software Engineering*, v. 2, p. 528–532.

Braga, R. T. V., Trindade, O., Branco, K. R. L. J. C. and Lee, J. (12 sep 2012). Incorporating certification in feature modelling of an unmanned aerial vehicle product line. . Association for Computing Machinery (ACM).

Fernandes, P. C. B., Guizzardi, R. S. S. and Guizzardi, G. (2011). Using Goal Modeling to Capture Competency Questions in Ontology-based Systems. *Journal of Information and Data Management*, v. 2, n. 3, p. 527.

Filho, J., Barais, O., Baudry, B., Viana, W. and Andrade, R. M. C. (2012). An approach for semantic enrichment of software product lines. In *ACM International Conference*

Proceeding Series.

Habli, I. and Kelly, T. (sep 2007). Challenges of Establishing a Software Product Line for an Aerospace Engine Monitoring System. In *11th International Software Product Line Conference (SPLC 2007)*. . IEEE.

ICAO (2011). Unmanned Aircraft Systems (UAS). *ICAO Cir 328, Unmanned Aircraft Systems (UAS)*

ISO (2018). Road vehicles -- Functional safety. . ISO, Geneva, Switzerland.

Kang, K. C., Kim, S., Lee, J., et al. (jan 1998). FORM: A feature-;oriented reuse method with domain-;specific reference architectures. *Annals of Software Engineering*, v. 5, n. 1, p. 143.

Kelly, T., Habli, I., Masiero, P. C., et al. (2016). Model-based safety analysis of software product lines. *International Journal of Embedded Systems*, v. 8, n. 5/6, p. 412.

Lee, K., Kang, K. C. and Lee, J. (2002). Concepts and guidelines of feature modeling for product line software engineering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 2319, n. April, p. 62–77.

Noy, N. F. and Hafner, C. D. (1997). The state of the art in ontology design: A survey and comparative review. *AI Magazine*, v. 18, n. 3, p. 53–74.

Oliveira, A. L., Braga, R. T. V., Masiero, P. C., et al. (2018). Variability Management in Safety-Critical Software Product Line Engineering. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. . Springer Verlag.

Pohl, K., Böckle, G. and Van der Linden, F. J. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer.

RTCA (2011). *DO-178C Software Considerations in Airborne Systems and Equipment Certification*. Radio Technical Commission for Aeronautics.

Villela, K., Silva, A., Vale, T. and De Almeida, E. S. (2014). A Survey on Software Variability Management Approaches. In *Proceedings of the 18th International Software Product Line Conference - Volume 1*. , SPLC '14. ACM.

Wohlin, C., Runeson, P., Hst, M., et al. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.