Combining MARTE-UML, SysML and CVL to Build Unmanned Aerial Vehicles

Paulo Gabriel Gadelha Queiroz

Departamento de Ciências Exatas e Naturais - DCEN
Universidade Federal Rural do Semi-Árido
Mossoró, Brazil
Email: pgabriel@ufersa.edu.br

Rosana Teresinha Vaccare Braga

Instituto de Ciências Matemáticas e de Computação - ICMC
Universidade de São Paulo
São Carlos, Brazil
Email: rtvb@icmc.usp.br

***Abstract*—Several methodologies have been proposed in the last decades to improve the quality of critical embedded systems and, at the same time, keep costs and schedule compatible with project plans. In particular for Unmanned Aerial Vehicles (UAV), approaches such as Product Line Engineering (PLE) and Model-Driven Engineering (MDE) offer an interesting solution to reduce development complexity and are being widely used in various academic research and industrial projects. This paper presents an approach combining PLE and MDE to develop families of Unmanned Aerial Vehicles. In this approach, we propose the use of SysML and MARTE UML profile to support requirements specification, design, validation, simulation and eventual code generation. Additionally, we propose the use of the Common Variability Language (CVL) to support the transformations of the generic product line models into specific product models, aiming at achieving a high degree of reuse. Additionally, this paper proposes a process to use the above mentioned modeling techniques to produce family models and a method to use these artifacts to generate product members. Finally, we illustrate the various concepts presented in the proposed methodology by means of a UAV case study.**

Keywords–Product Line; Model-Driven Development; Safety-Critical Systems.

I. Introduction

Embedded Systems are components integrating software and hardware jointly and specifically designed to provide given functionalities [1]. Safety-Critical Embedded Systems (SCES), in particular, are embedded systems whose failure could result in loss of lives or on significant environmental or property damage. SCES are common in medical devices applications, aircraft flight control systems, weapons, and nuclear systems. Aircraft flight control systems, for example, must present failure rates as low as a serious fault per 10^8 flight hours [2] and other complex constraints and requirements like cost-effectiveness, time to market, fast evolving environment, reliability, security, availability, criticality, reactivity, autonomy, robustness, and scalability [3], which impose overhead costs on the development. In the SCES domain, we focus on Unmanned aerial vehicles (UAV), which can be defined as airplanes that fly without the need of a human pilot, accomplishing a pre-established mission.

The coming generations of SCES, like UAVs, must meet the new expectations created by hardware evolution like the increase in computational power of processors and the corresponding decrease in size and cost that lead to the increase of users expectations for new functionalities and have allowed moving more and more functionality to software [4]. Therefore, to overcome these challenges and to fulfill the requirements and constraints mentioned above, we need new efficient and flexible development methodologies and tools that can reduce UAV production complexity.

The use of Product Line Engineering (PLE) [5] has proven to be a good alternative to reduce system costs and time to market, as well as to increase system reliability through the assembling of reusable and extensively tested resources. Model-Driven Engineering (MDE) [6] is also used in this context, producing models in higher abstraction levels and allowing automatic generation of products through model transformations.

The motivations for this work have arisen after the creation of a Product Line (PL) workgroup in the National Institute of Science and Technology - Critical Embedded Systems (INCT-SEC) [7] project, whose goal was to create methodologies and tools to develop, among others, families of UAVs. The first family, called Tiriba, was developed by the AGX Company [8] in partnership with INCT-SEC. During our participation in this workgroup we performed a systematic review of the literature to find gaps in existing methodologies and approaches that combine MDE and PLE to develop SCES, as well as a study using three other UAV existing examples [9][10][11] to build a family of UAVs and validate our approach. These studies culminated with the approach proposed in this paper, whose main goal is to build a family of UAV using a combination of both PLE and MDE techniques. The novelties of the proposed approach are the use of MDE in both Domain and Application Engineering and the management of both software and hardware variabilities, accompanied by Verification and Validation (V&V) activities during the whole cycle. For an effective use of MDE, we propose the use of a subset of the Systems Modeling Language (SysML) [12] and the UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) [13] to enable model transformations in the Domain Engineering (DE) phase. We also propose the use of the Common Variability Language (CVL) [14] to manage system variabilities and enable model transformations during the Application Engineering (AP) phase. Finally, we illustrate the various concepts present in the proposed approach by means of a UAV family case study. It is worth to mention that this approach is an extension of the work presented in [15], with the addition of CVL to manage variabilities, the safety analysis activity and the evolution of the case study.

The rest of this paper is organized as follows: Section II presents a background summary of Product Line Engineering and variability management; Section III summarizes related works; Section IV presents the proposed approach; Section V illustrates the proposed approach by means of a UAV product line case study; lastly, Section VI presents the conclusions of this paper.

II. Background

Product Line Engineering is an approach that enables organizations to develop, deliver and evolve an entire Product Line portfolio, through each stage of the development life cycle, with much higher degrees of efficiency compared to developing single systems [16]. The products of a PL differ from each other in terms of features, which are user-visible aspects or characteristics of a software system or systems [17]. As expected, the costs, in terms of time and money spent, to build a PL is higher in comparison with the costs to build a single system, because among other things PLE is done in two stages: Domain Engineering (DE), which is the development of a series of generic artifacts to the PL; and Application Engineering (AE), in which the application engineer uses the artifacts developed in DE to assemble products of the line, known as members.

During DE, a general architecture for the PL is defined, from which various products can be generated. Despite the higher cost, Weiss and Lai [18] claim that the construction of a PL is justified if at least three systems generated from the PL are derived. Since UAV are often manufactured, distributed in large scale and present significant variability in terms of hardware and applications, it can be expected that the use of PLE is advantageous to them.

On the other hand, in Model-Driven Engineering, the software complexity concentrates on high level models and not in the code, which can be automatically generated from the models. Furthermore, system quality can be improved with the use of V&V methods [19]. According to model-based approaches, models become part of the final product and most of the development complexity shall belong to the transformations that should be used to automatically or semi-automatically produce code. To successfully use MDE techniques to model a UAV PL, we propose the use of SysML, MARTE and CVL.

SysML [12] is a general-purpose modeling language for systems engineering applications, which reuses a subset of the Unified Modeling Language (UML) [20] and provides additional extensions. SysML supports the specification, analysis, design, verification, and validation of a broad range of complex systems and is used to model a wide range of industrial and academic systems [21]. As SysML is an UML extension, there is a compatibility of tools and concepts, which can reduce the learning time. We also want to propose an approach that can be adopted using free tools, which are abundant for UML-based languages.

MARTE [13] is an UML profile that provides capabilities for model-driven development of Real Time and Embedded Systems (RTES). It provides support for specification, design, and verification/validation stages [13]. MARTE is also used to model a wide range of industrial and academic systems [22]. We adopt it in our approach because UAVs have many real time constraints that need to be checked by model simulation in early development stages, to improve product quality.

Even though some authors consider MARTE and SysML profiles incompatible, by using the MADES methodology [3] recommendation we can avoid conflicts related to the two profiles by not mixing SysML and MARTE concepts in the same diagram, but instead focusing on a refinement scheme. Therefore, as presented later, SysML is used for initial requirements and functional description, while MARTE is utilized for the enriched modeling of the global functionality and execution platform/software modeling.

Another resource that can be useful to improve the application of MDE techniques to build the UAV PL is CVL [14], which is a separate and generic language to define variabilities. CVL semantics are defined as a transformation of an original model (e.g., a product line model) into a configured, new product model. CVL combines user-centric feature diagrams with an automation-centric approach to the production of product models. In CVL, the focus is on specifying variability in a model separate from the base product line models. A base model is an instance of any metamodel conforming to Meta Object Facility (MOF) [23]. The base models are produced in domain engineering in our case. There may be several variability models applied to the same base product line model and the base model is unaware of the variability models (there are only links from the CVL model to the base model). Several product resolutions can apply to the same variability model. CVL is executable, i.e., after specifying the resolution of variabilities, a CVL tool can automatically derive the specific product model.

The core concept of CVL is substitution. Models are assumed to consist of model elements in terms of objects that are related by means of references. The CVL model points out model elements of the base PL model and defines how these model elements shall be manipulated to yield a new product model. There are three kinds of substitution: value substitution, reference substitution and fragment substitution. A substitution replaces base model elements named as placement by base model elements named as replacement [14]. CVL can represent variabilities through the concepts of Variation point, Substitution, Existence, Value assignment, Variability specification, and Choice, among others.

III. Related works

While there is a large number of researches who make use of either PLE or MDE for safety-critical embedded systems, due to space limitations, it is not possible here to give an exhaustive description, so we only provide a brief summary of works that combine PLE and MDE to build safety-critical embedded systems, similarly to the approach proposed in this work.

The work presented by Polzer et al. [24] is concerned with variability in control systems software, where a model-based PL engineering process using Rapid Control Prototyping system is combined with MDE techniques. The authors modularize the components parameterization in a separate setup, which is isolated from the model that defines the behavior of the controller. Simulink [25] and Pure::variants [26] are used for modeling and automatic code generation. It is observed that this work is done with proprietary tools and modeling techniques like Matlab building blocks that although efficient for the project description, are not ideal for requirements modeling and communication with the final user, which goes against the purpose of this paper.

Regarding the development of UAV product lines, there are approaches such as Product Line on Critical Embedded Systems (ProLiCES) [27] and SysML-based Product Line Approach for Embedded Systems (SyMPLES) [28]. Even though they were not defined for UAV, the authors used a UAV case study to illustrate their approaches. ProLiCES creates a parallel path in the process to handle the PL domain engineering and also proposes the use of Matlab/Simulink as a Model-Driven Development (MDD) technique, which limits requirements analysis and concentrates the MDD only in one step of the process. SyMPLES is an approach for PL application in embedded systems through the extension of SysML language to include variability together with a development process, but in this study the authors do not distinguish between the characteristics of hardware and software and focus on the use of SysML for the architecture description.

Svendsen et al. [29] present a case study for creating a PL for the train signaling domain. The Train Control Language (TCL) is a Domain-Specific Language that automates the production of source code for computer-controlled train stations, also using CVL. However, their approach presents just the variability management through CVL, which consists of a portion of the system product line development process.

In the work presented by Haber et al. [30] the authors focus on variability management in all development phases using Matlab/Simulink. They propose a modular variability modeling approach based on the concept of delta modeling. A functional variant is described by a delta encapsulating a set of modifications. A sequence of deltas can be applied to a core product to derive the desired variant. The authors illustrate the approach by presenting a prototypical implementation.

Finally, we highlight the Cardiac Pacemaker PL described by Huhn and Bessling [31], where they present how to specify the PL and its products by means of CVL. CVL enforces a strict structuring of the product models (done in SCADE) that reflects the substitution concepts used to describe variability.

The approach we propose in this paper is different from the above mentioned related works, as it focuses on the PL definition, modeling both hardware and software variabilities. We propose the use of MDE, like automatic generation of hardware descriptions and embedded software from high level models, for rapid design and specification of SCES. Furthermore, we propose the use of free tools for MDE, the use of UML as it is an extensively used modeling language and the use of CVL to model variability.

IV. PL APPROACH

Figure 1 illustrates our proposed Product Line approach. Notice that the approach addresses both hardware and software variability with its underlying requirement dependencies. To reduce both domain and application engineering complexity, we propose the use of UML based models, in particular SysML and MARTE. Despite being the most widely used modeling language, UML is generally easier to understand than Matlab blocks.

The strengths of the proposed methodology are the use of MDE in both Domain and Application Engineering phases, with focus on model-to-model transformations in requirements, analysis and design activities, especially for the purpose of modelling and generating application variants. This is different from most of the PL methodologies, which focus on model-to-text transformations just in the Application Engineering phase, through the use of application generators. The use of MDE has also the advantage to promote the possibility to use Model-Based Test [32] in early design stages, which can substantially reduce the V&V costs and effort [33]. Safety analysis [34] techniques is also recommended in early design stages, especially for certification purposes, but this is out of the scope of this paper.

As seen in Figure 1, the approach is divided into two interdependent phases, Domain Engineering and Application Engineering, which are common in consolidated methods like the Framework for Software Product Line Practice [16] and the PLUS method [35]. During the DE phase, the high level system models are carried out using SysML and CVL, which are exemplified later in Section V. After the system PL specification (user requirements, specification and related hardware/software variability specification), underlying model transformations (model-to-model and model-to-text transformations) are used to produce models for the subsequent design phases including MARTE profile. The next design phases include verification, hardware descriptions of modeled targeted architecture and generation of platform specific embedded software from platform independent software specifications. For implementing model transformations in the case study, we use the Eclipse Modeling Platform (EMF) [36]; the Papyrus [37] modeling tool, which is a UML modeler that enables model transformations, code generation and validation; and the CVL Eclipse Plug-in [38] as the engine for the transformations of product line models into specific product models. The proposed approach is not limited to these tools, therefore the choice of the modeling tool is up to the user, the only requirement is to support MARTE and SysML metamodels.

Another important factor to be noted is that in the UAV domain, hardware variability could impact directly on software requirements and vice versa. For example, consider the following system requirement: *the system should allow the user to choose between broadcasting the images to the ground control station in real time or to recording a video (in flash memory, for example)*. In that case, the UAV hardware must include a camera. Moreover, for each new sensor added, their corresponding software drivers must also be added. Another highlight is the continuous feedback in the artifacts repository, in which we can store any kind of artifact from both hardware or software types. As a repository to store hardware artifacts, we refer in a logical level, to a hardware models repository (the same repository to store software artifacts). This feedback can come from updates in DE or from new different requirements elucidated from new members modeled in the AE. The feedback is represented by both dashed arrows and double-headed arrows.

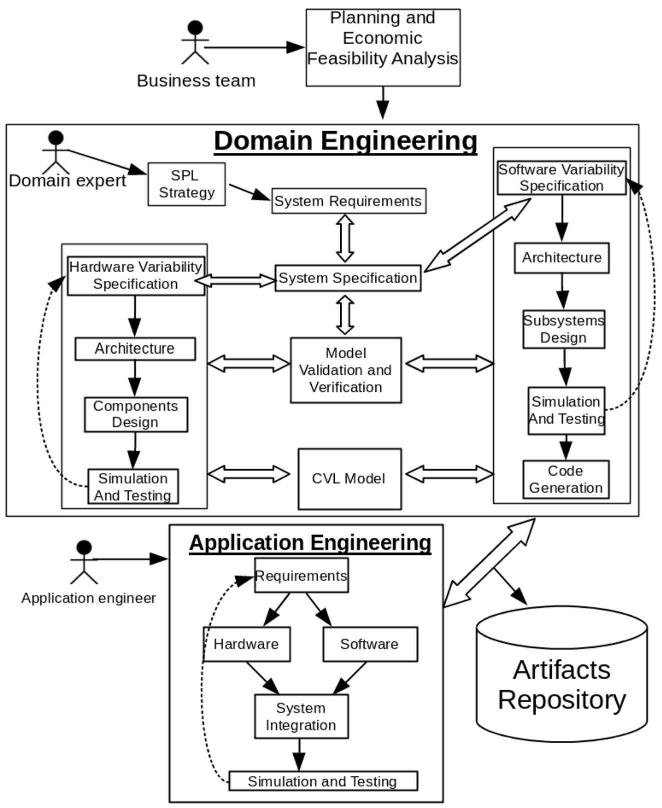


Figure 1: Overview of the proposed approach.

A. Domain Engineering

Before the Domain Engineering takes place, a business team performs an economic feasibility analysis of the PL, which will indicate whether or not it is worth to be developed. If the PL is feasible, then we start the Domain Engineering by modeling requirements in the system abstraction level, as detailed below. It is out of the scope of this work to propose domain analysis techniques, as they can be easily found in the literature. So, existing techniques such as those mentioned in the survey by Prieto-Diaz and Arango [39] can be used.

The Domain Engineering is performed by the domain expert, who should first define the PL strategy, i.e., he must decide whether to use a proactive or reactive approach [40]. Regardless of the strategy chosen, to model variabilities we propose the use of CVL. Since CVL replaces values and sets of model elements, by executing CVL we can add, remove or replace functionality. To use CVL, according to Svendsen et al. [29] the domain expert has three options for choosing a base model: the first is a model with maximum set of features included, meaning a complete model where CVL can remove features to get a specific product model; the second is a model with a minimum set of features included in the model itself, and other fragments in other library models, then the product models will be generated by adding features to the base model; and the third is to choose a base model that has neither maximum nor minimum, but somewhere in between, so this base model can be, for instance, the base model that is most similar to the majority of the product models, or a base model that is tailored for teaching purposes.

CVL proposes a model with two parts: the Feature Specification Layer (FSL) and the Product Realization Layer (PRL). The FSL resembles a feature diagram [17], while the PRL connects the FSL to the base model, for example by substitutions. We also suggest that domain experts develop the FSL incrementally: first, create a high level system FSL, then add the software and the hardware features. Hardware variability management should concern the impact evaluation of hardware variabilities on software requirements, in the same way that software variability management should concern the impact evaluation of software variabilities in hardware requirements. To manage this impact, the dependencies between hardware and software features should be defined.

As we propose the use of CVL in conjunction with a subset of MARTE and SysML models to describe the PL, the domain expert should prepare a PRL that corresponds to each SysML and MARTE models, so that after the variability resolution, all models created for the PL are automatically adapted for each product.

To complement the system requirements definition, a SysML requirements diagram should be created, where distinguishing between functional and non-functional requirements is recommended. As illustrated in Figure 1, the artifact repository is updated during the PL life cycle for both domain experts and application engineers. So, if concrete products have new requirements not covered by the PL, the requirements diagram can be further reviewed to include them.

Following the system requirements activity, we proceed to the system specification activity, where we produce use case scenarios and a system domain model. The use case is described using traditional UML Use Case Diagrams. The system domain model can be modeled using a class diagram, in which the concepts are represented by pseudo-classes. These two modeling concepts are strongly related to the functional high level specification described subsequently. While the use case is needed to obtain a SysML block diagram, as explained below, the domain model is used to communicate with domain experts for a better understanding about the domain, for validating the specification and for a future definition of a domain specific language by means of a UML profile, for example.

To finalize this system initial description, each use case is converted into a SysML block (or internal block), for example by applying the MADES methodology [3], with the difference that a mandatory use case is converted to a mandatory block, an optional use case is converted to an optional block, and an alternative use case is converted to an alternative block. After including all the developed artifacts in the repository, we can continue the PL development by going to hardware or software abstract levels or even to both in parallel.

Following this initial system specification, the development can evolve into two parallel paths, as illustrated in Figure 1. The first path starts at hardware specification, architecture definition, design of the components and simulation. The second path goes through software variability specification and management, architecture definition, subsystems design, simulation, testing, and code generation.

The designer can move to the partitioning of the system in question: depending upon the requirements and resources in hand, he or she can determine which part of the system needs to be implemented in hardware or software. It is possible, although it could substantively increase SPL costs, to improve safety by implementing system features in a redundant way, i.e., whenever possible, to provide features implementations in both hardware and software. Thus, it becomes part of the Application Engineering to decide if the features implementation component should be integrated in the product by software or hardware.

Since the proposed approach focuses on UAV, V&V activities should be executed in each stage. It is important to notice that on hardware and software paths a more detailed specification takes place by the eventual allocation with schedulability and underlying model transformations (model-to-model and model-to-text transformations) that are used to bridge the gap between these abstract design models and subsequent design phases. These phases include verification, hardware descriptions of modelled target architecture and generation of platform specific embedded software from independent architectural software specifications.

For a description of the different steps related to each design level by means of MARTE concepts, see the work of Quadri et al. [3], which can be adapted to this approach by creating a CVL model for the base models representing the hardware and software specification, architectural definition, components and subsystems design and simulation. It is also important to perform validation activities in every model to ensure they correspond to the requirements and are traceable to each other. At the end of this phase, our repository contains all the artifacts and the domain engineering is ended.

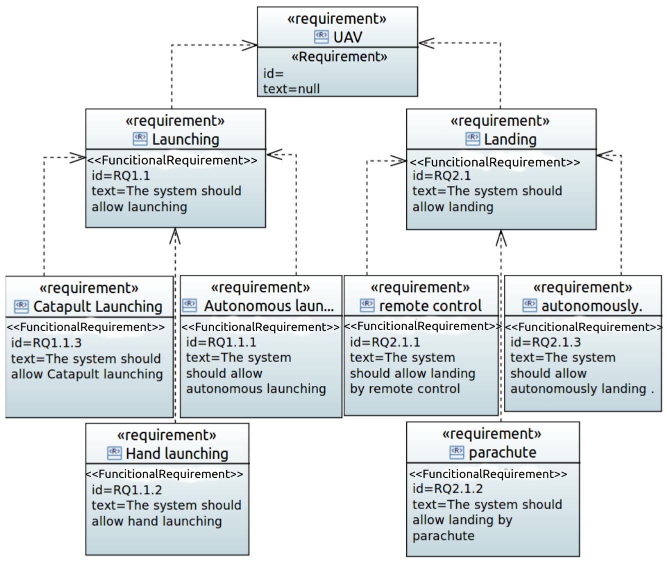
B. Application Engineering

Application Engineering corresponds to configuring a product by assembling reusable artifacts from the repository. This step is the responsibility of the application engineer, who elicits the particular system requirements. By using our approach, the application engineering phase is simplified and reduced to the definition of the resolution model, which consists of selecting the desired features for the PL member. So, the application engineer can choose which substitutions to execute, and then execute the CVL model that will generate specific products (i.e., specific models). To conclude this step, it is necessary to conduct simulation and testing also on the target system to validate it.

V. Case Study

To illustrate the use of the proposed methodology, we present the initial development of a UAV PL. Through this example, we aim to show the power of CVL to manage hardware and software variabilities. We assume that the PL economic feasibility analysis indicated it is worth to be developed. The domain expert defined the following strategies: reactive approach with the base model with maximum set of features. We have chosen Tiriba as a starting point, but intend to include other UAVs in future works.

In Figure 2, we illustrate part of the SysML requirements diagram with the maximum set of features. The next step is to create the CVL model, which comprises FSL and PRL, for this base model. For the management of both hardware and


«requirement»
UAV
«Requirement»
id=
text=null
«requirement»
Launching
<<FuncitionalRequirement>>
id=RQ1.1
text=The system should allow launching
«requirement»
Landing
<<FuncitionalRequirement>>
id=RQ2.1
text=The system should allow landing
«requirement»
Catapult Launching
<<FuncitionalRequirement>>
id=RQ1.1.3
text=The system should allow Catapult launching
«requirement»
Autonomous laun...
<<FuncitionalRequirement>>
id=RQ1.1.1
text=The system should allow autonomous launching
«requirement»
remote control
<<FuncitionalRequirement>>
id=RQ2.1.1
text=The system should allow landing by remote control
«requirement»
autonomously.
<<FuncitionalRequirement>>
id=RQ2.1.3
text=The system should allow autonomously landing .
«requirement»
Hand launching
<<FuncitionalRequirement>>
id=RQ1.1.2
text=The system should allow hand launching
«requirement»
parachute
<<FuncitionalRequirement>>
id=RQ2.1.2
text=The system should allow landing by parachute
Figure 2: Part of the SysML requirements diagram for the UAV Product Line.

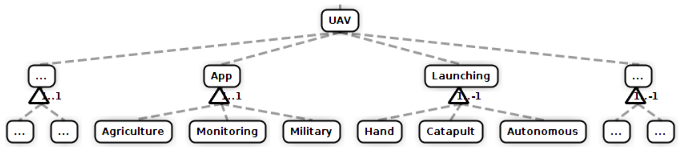


Figure 3: Part of the system high level FSL for the UAV Product Line.

software features, we should create a hardware and software FSL, which are modelled from the system high level feature diagram, as illustrated in Figure 3. To finish this first part, we define the PRL, which connects the feature specification layer to the base model and the substitutions, like illustrated in Figure 4 through an architectural view. Observe that the base model has the maximum set of features, thus a subtractive strategy is used for most parts, and sometimes a substitution.

Supposing that the same process is done for the other models proposed and all the models are stored in a repository, all the application engineer needs to do is to create the resolution model, by selecting the required features for the product. A possible resolution model with the positive choices for *Autonomous Navigation*, *Manual Control*, *Agrochemical Sprayer* and *Video Camera* is illustrated in Figure 5. After executing the CVL engine, the PRL is transformed into the resulting product model presented in Figure 6.

VI. Conclusions and Future Works

This paper aimed to present an approach for UAV product lines modeling with the use of MDE techniques in both Domain and Application Engineering, as well as software and hardware variability management. To fulfill this objective we have used a subset of UML profiles like SysML, MARTE in combination with the CVL. The use of the proposed approach in the UAV domain can bring the benefits of PL and MDE techniques, such as reducing system costs and time-to-market

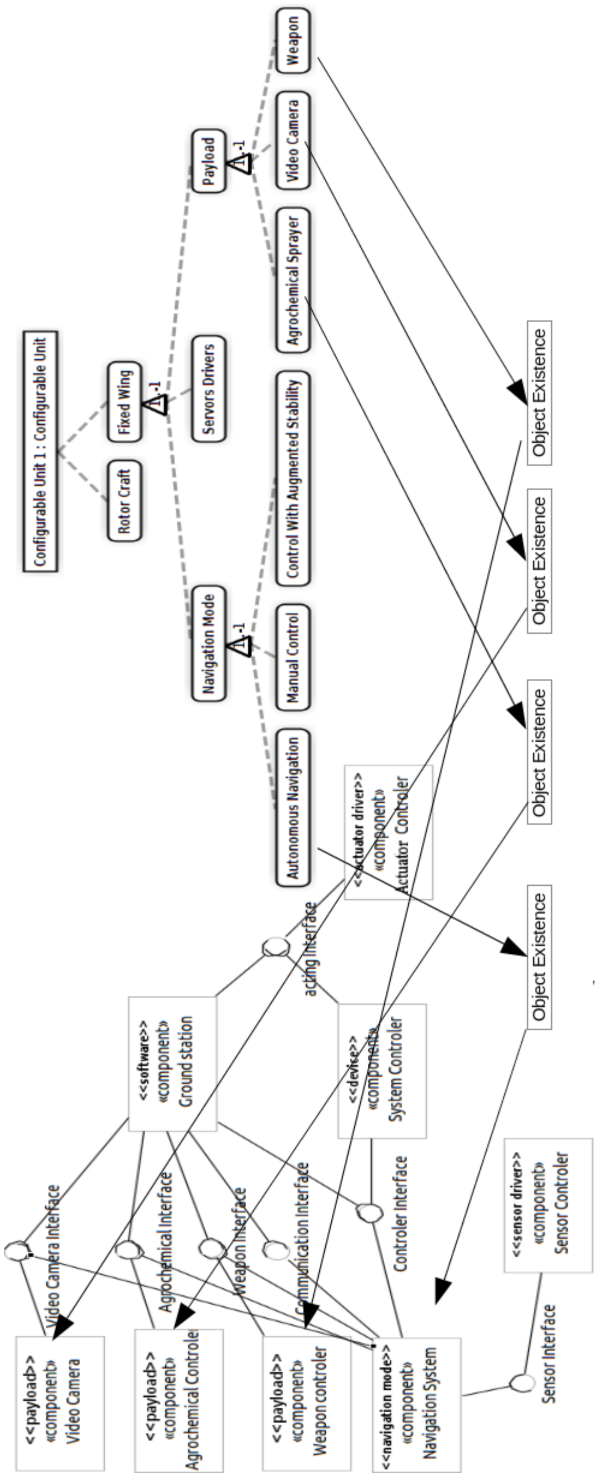

Configurable Unit 1 : Configurable Unit
Rotor Craft
Fixed Wing
Servors Drivers
Navigation Mode
Payload
Autonomous Navigation
Manual Control
Control With Augmented Stability
Agrochemical Sprayer
Video Camera
Weapon
Object Existence
Object Existence
Object Existence
Object Existence
<<actuator driver>> «component» Actuator Controler
acting Interface
<<software>> «component» Ground station
<<device>> «component» System Controller
<<sensor driver>> «component» Sensor Controler
Video Camera Interface
Agrochemical Interface
Weapon Interface
Communication Interface
Controler Interface
Sensor Interface
<<payload>> «component» Video Camera
<<payload>> «component» Agrochemical Controler
<<payload>> «component» Weapon controler
<<navigation mode>> «component» Navigation System

Figure 4: Part of the PRL for the UAV Product Line.

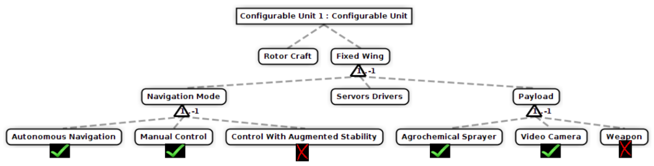


Figure 5: Part of the UAV resolution model.

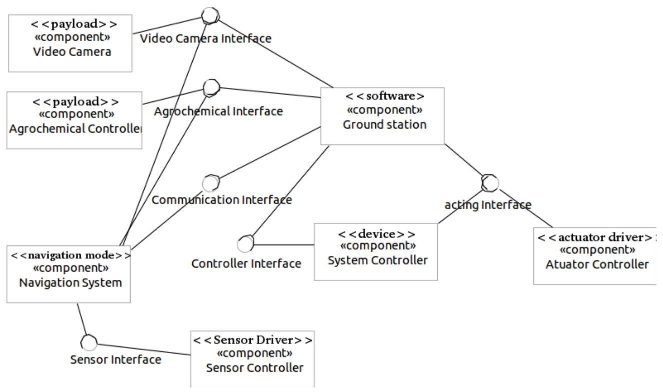

<<payload>> «component» Video Camera
Video Camera Interface
<<payload>> «component» Agrochemical Controller
Agrochemical Interface
<<software>> «component» Ground station
Communication Interface
acting Interface
<<navigation mode>> «component» Navigation System
Controller Interface
<<device>> «component» System Controller
<<actuator driver>> «component» Atuator Controller
Sensor Interface
<<Sensor Driver>> «component» Sensor Controller

Figure 6: Part of the UAV product model after processing the variabilities.

and increasing system reliability. Finally, the most important steps, models and concepts from the proposed approach have been illustrated by a UAV product line case study. The main limitation of our approach is the lack of definition of a UAV UML profile to improve model-to-code transformation. Therefore, for future work we propose to define a UAV UML profile and to evaluate the proposed approach by means of an experiment to compare our approach with others in terms of efficiency and usability.

Acknowledgments

The authors would like to thank CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) for financial support received during the development of this work.

REFERENCES

[1] J. Sifakis, “Embedded systems - challenges and work directions,” in Principles of Distributed Systems, 8th International Conference, Grenoble, France, ser. Lecture Notes in Computer Science, T. Higashino, Ed., vol. 3544. Springer, Dec. 2004, pp. 184–185.

[2] I. Moir, Civil Avionics Systems, ser. Aerospace Series (PEP). John Wiley Sons Ltd, 2006.

[3] I. R. Quadri, A. Sadovykh, and L. S. Indrusiak, “MADES: a SysML/MARTE high level methodology for real-time and embedded systems,” in ERTS2 2012: Embedded Real Time Software and Systems, Feb. 2012, pp. 1–10.

[4] J. R. Burch, R. Passerone, and A. L. Sangiovanni-Vincentelli, “Using multiple levels of abstractions in embedded software design,” in Embedded Software, ser. Lecture Notes in Computer Science, T. A. Henzinger and C. M. Kirsch, Eds. Springer Berlin Heidelberg, Sep. 2001, vol. 2211, pp. 324–343. [Online]. Available: http://dx.doi.org/10.1007/3-540-45449-7_23 [retrieved: August, 2014]

[5] K. Pohl, G. Böckle, and F. J. v. d. Linden, Software Product Line Engineering: Foundations, Principles and Techniques. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[6] S. Kent, "Model driven engineering," in Integrated Formal Methods, ser. Lecture Notes in Computer Science, M. Butler, L. Petre, and K. Sere, Eds. Springer Berlin Heidelberg, Apr. 2002, vol. 2335, pp. 286–298. [Online]. Available: http://dx.doi.org/10.1007/3-540-47884-1_16 [retrieved: August, 2014]

[7] INCT-SEC, "Sistemas Embarcados Críticos: aplicações em segurança e agricultura," 2008. [Online]. Available: http://www.inct-sec.org [retrieved: August, 2014]

[8] AGX Tecnologia Ltda, 2014. [Online]. Available: www.agx.com.br [retrieved: August, 2014]

[9] GISA-Grupo de Interesse em Sisvants e Aplicaes, 2014. [Online]. Available: http://gisa.icmc.usp.br/site/ [retrieved: August, 2014]

[10] SLUGS-Santa Cruz Low-cost UAV GNC System , 2014. [Online]. Available: http://slugsuav.soe.ucsc.edu/ [retrieved: August, 2014]

[11] Ardupilot , 2014. [Online]. Available: http://ardupilot.com/ [retrieved: August, 2014]

[12] Object Management Group, OMG Systems Modeling Language (OMG SysML), V1.3, 2012.

[13] Object Management Group*, UML Profile for MARTE (Modeling and Analysis of Real-Time and Embedded Systems) 1.1, 2011, OMG doc. http://www.omg.org/spec/MARTE/1.1/ [retrieved: August, 2014].

[14] O. Haugen, A. Wasowski, and K. Czarnecki, "Cvl: Common variability language," in Proceedings of the 16th International Software Product Line Conference - Volume 2, ser. SPLC '12. New York, NY, USA: ACM, 2012, pp. 266–267. [Online]. Available: http://doi.acm.org/10.1145/2364412.2364462

[15] P. G. G. Queiroz and R. T. V. Braga, "A critical embedded system product line model-based approach," in SEKE-26: Proceedings of the The 26th International Conference on Software Engineering and Knowledge Engineering. London, UK: Springer, Jul. 2014, pp. 71–75.

[16] Northrop, L. M. et al., "A Framework for Software Product Line Practice, Version 5.0," 2009. [Online]. Available: http://www.sei.cmu.edu/productlines/framework.html [retrieved: August, 2014]

[17] K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," IEEE Software, vol. 19, no. 4, Aug. 2002, pp. 58–65.

[18] D. M. Weiss and C. T. R. Lai, Software Product-line Engineering: A Family-based Software Development Process. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[19] L. Belategi, G. Sagardui, and L. Etxeberria, "Model based analysis process for embedded software product lines," in Proceedings of the 2011 International Conference on Software and Systems Process, ser. ICSSP '11. New York, NY, USA: ACM, May. 2011, pp. 53–62. [Online]. Available: http://doi.acm.org/10.1145/1987875.1987886 [retrieved: August, 2014]

[20] O. M. Group, "OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2," Tech. Rep., nov 2007. [Online]. Available: http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF [retrieved: August, 2014]

[21] P. Baker, S. Loh, and F. Weil, "Model-Driven Engineering in a Large Industrial Context - Motorola Case Study," in Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems, ser. MoDELS'05. Berlin, Heidelberg: Springer-Verlag, Oct. 2005, pp. 476–491.

[22] M. Z. Iqbal, A. Arcuri, and L. Briand, "Environment modeling with uml/marte to support black-box system testing for real-time embedded systems: Methodology and industrial case studies," in Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems: Part I, ser. MODELS'10. Berlin, Heidelberg: Springer-Verlag, Oct. 2010, pp. 286–300. [Online]. Available: http://dl.acm.org/citation.cfm?id=1926458.1926486 [retrieved: August, 2014]

[23] omg, Meta Object Facility (MOF) Core Specification Version 2.0, 2006. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[24] A. Polzer, S. Kowalewski, and G. Botterweck, "Applying software product line techniques in model-based embedded systems engineering," in Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2009), Workshop at the 31st International Conference on Software Engineering (ICSE 2009), vol. 0. IEEE Computer Society, May May. 2009, pp. 2–10.

[25] O. Beucher, MATLAB und Simulink (Scientific Computing). Pearson Studium, 08 2006.

[26] Pure Systems, "pure::variants," 2012.

[27] Braga, R. T. V. et al., "The prolices approach to develop product lines for safety-critical embedded systems and its application to the unmanned aerial vehicles domain." CLEI Electron. J., vol. 15, no. 2, May 2012, pp. 1–13. [Online]. Available: http://dblp.uni-trier.de/db/journals/cleiej/cleiej15.html#BragaBJMNB12 [retrieved: August, 2014]

[28] R. F. Silva, V. H. Fragal, E. A. de Oliveira Junior, I. M. de Souza Gimenes, and F. Oquendo, "SyMPLES - A SysML-based Approach for Developing Embedded Systems Software Product Lines." in ICEIS (2), S. Hammoudi, L. A. Maciaszek, J. Cordeiro, and J. L. G. Dietz, Eds. SciTePress, Jul. 2013, pp. 257–264. [Online]. Available: http://dblp.uni-trier.de/db/conf/iceis/iceis2013-2.html#SilvaFJGO13 [retrieved: August, 2014]

[29] Svendsen, A. et al., "Developing a software product line for train control: A case study of cvl." in SPLC, ser. Lecture Notes in Computer Science, J. Bosch and J. Lee, Eds., vol. 6287. Springer, Sep. 2010, pp. 106–120. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15579-6_8 [retrieved: August, 2014]

[30] Haber, A. et al., "First-class variability modeling in matlab/simulink," in Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems, ser. VaMoS '13. New York, NY, USA: ACM, Jan. 2013, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2430502.2430508 [retrieved: August, 2014]

[31] S. Blessing and M. Huhn, "Formal Safety Analysis and Verification in the Model Driven Development of a Pacemaker Product Line." in MBEES, H. Giese, M. Huhn, J. Phillips, and B. Schtz, Eds. fortiss GmbH, Mnchen, Feb. 2012, pp. 133–144. [Online]. Available: http://dblp.uni-trier.de/db/conf/mbees/mbees2012.html#BlessingH12 [retrieved: August, 2014]]

[32] M. Timmer, H. Brinksma, and M. I. A. Stoelinga, "Model-based testing," in Software and Systems Safety: Specification and Verification, ser. NATO Science for Peace and Security Series D: Information and Communication Security, M. Broy, C. Leuxner, and C. A. R. Hoare, Eds. Amsterdam: IOS Press, April 2011, vol. 30, pp. 1–32.

[33] Heimdahl, Mats Per Erik, "Safety and software intensive systems: Challenges old and new." in FOSE, L. C. Briand and A. L. Wolf, Eds., 2007, pp. 137–152.

[34] J. Liu, J. Dehlinger, and R. Lutz, "Safety analysis of software product lines using state-based modeling," in 16th IEEE International Symposium on Software Reliability Engineering, 2005. ISSRE 2005, pp. 10–30.

[35] H. Gomaa, Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Redwood City, CA, USA: Addison Wesley Longman, 2004.

[36] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework 2.0, 2nd ed. Addison-Wesley Professional, 2009.

[37] "Open source tool for UML modeling," 2011, http://www.papyrusuml.org/ [retrieved: August, 2014].

[38] SINTEF, "Cvl tool from sintef," 2012. [Online]. Available: http://www.omgwiki.org/variability/doku.php/doku.php?id=cvl_tool_from_sintef [retrieved: August, 2014]

[39] R. Prieto-Diaz and G. Arango, Domain Analysis and Software Systems Modeling. IEEE Press, 1991.

[40] C. W. Krueger, "Variation management for software production lines," in Proceedings of the Second International Conference on Software Product Lines, ser. SPLC 2. London, UK, UK: Springer-Verlag, Aug. 2002, pp. 37–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=645882.672255 [retrieved: August, 2014]