Abstract—An Unmanned Aerial Vehicle (UAV) is controlled by an embedded system that works autonomously by following commands stored in an internal computer. An UAV contains elements that may vary according to the domain application. Thus, such elements might be managed by the Software Product Line (PL) approach. UAVs and PLs use different tools for managing their specifications and models. Therefore, this paper presents an approach for mapping PL features to UAV models. Such an approach is supported by a developed tool, the SimulinkImport. Specific elements of a mini-UAV were used as a proof of concept and to illustrate this approach.

Keywords—Embedded Systems, Features, Mini-UAV, Simulink, Software Product Line, Unmanned Aerial Vehicle.

I. INTRODUCTION

An Unmanned Aerial Vehicle (UAV) is a critical embedded system in which a software communicates to hardware devices and provides high availability. UAV requirements must be carefully specified as there are several constraints including: memory and processing limitations, reduced energy consumption, specific architecture components, and tasks that must be executed parallelly and in real time [2].

The UAV development produces components that may vary according to the domain application and may be managed by means of the Software Product Line (PL) approach. Such an approach aims at developing software systematically to generate specific products by reusing a central infrastructure [10]. Similar products can be developed from a common core asset.

The application of the PL approach to the UAV domain is essential as the approach allows generating similar UAV models for a product by increasing their use in several applications. However, UAV model specifications and PL feature management are carried out with different approaches, which makes it difficult to develop PLs for UAVs. Simulink [11] is one of the most used tools to represent UAVs, whereas pure::variants [15] can be used to represent PL artifacts.

Pure::variants connector is currently a tool that can be used to support the management of functional variants in Simulink models. It fulfills the lack of variability description and allows variant management integration between pure::variants and Simulink. Pure::variants connector does not import full Simulink models to pure::variants, thus it only takes into account its library particular Simulink elements. Therefore, it only supports developers to link and test variants between tools.

This paper presents an approach for mapping PL features to Simulink model elements. In order to support such an approach, it was developed the SimulinkImport tool. It imports a Simulink model and its important subsystems allowing the mapping between PL features and regular Simulink elements. Thus, SimulinkImport can help developers to instantiate Simulink models for the representation of negative variabilities leading to code generation. An example using a mini-UAV is presented as a proof of concept, as well as to illustrate the application of such an approach.

This paper is organized as follows: Section II characterizes essential concepts of UAV, PL, and PL for UAVs; Section III presents the proposed approach for mapping PL features to an UAV Simulink model; Section IV discusses related work; and Section V presents conclusion and directions for future work.

II. BACKGROUND

This section presents fundamental concepts with regard to UAVs, PLs, and the application of the PL approach to the UAV domain. These concepts form the basis of the proposed approach (Section III).

A. Unmanned Aerial Vehicle (UAV)

Embedded systems are products created by the integration of computational modules that realize specific tasks, such as: cell phones, washing and drying machines, cars, and airplanes. An UAV is a critical embedded system as it must handle potential failures at runtime. Amongst important UAV quality attributes, reliability and performance are the most rigorous. Failures in this kind of system result in risks for human beings and high cost products [16].

The UAV development usually takes into account commercialization standards and methodologies. The use of models and code generators for embedded systems development is increasing. Simulink is a graphical modeling tool that allows the specification, modeling and simulation of basic block-based systems. Simulink is certified and referenced by several standards, such as DO-178B [6], which allows the UAV formal software development.

Buschmann et al. [4] developed a mini-UAV using a control system structure that includes actuator models, sensors and wind as presented in Figure 1. The control system consists of a Damper system, a Basic Controller to stabilize...
the mini-UAV’s height, and an Autopilot to control the mini-UAV’s course, height, and speed. Autonomous Navigation aims at defining the navigation points and mission accomplishment. This dynamic model allows controlling the mini-UAV in lower levels if a higher level problem occurs. In addition, it is simple to adjust parameters for a larger UAV.

Based on a complete mini-UAV model, it is possible to generate source code for a specific hardware [7], which is facilitated by using Simulink. The Simulink model can generate executable code for a specific machine by means of the Real Time Workshop Embedded Coder [12] tool.

B. Software Product Line (PL)

The PL approach aims at developing products systematically based on a family of products to generate specific products by means of the reusing of artifacts. A family of products is a set of similar products that share the same architecture and present certain differences. Such differences are represented as variabilities [10], [13]. A PL includes artifacts to define requirements, the architecture and its components, and system testing. These artifacts are created during the PL Engineering activities, which are:

- Domain Engineering: develops the central infrastructure that is composed of components and their constraints. Variation points that allow the infrastructure reusing are identified;
- Application Engineering: develops PL products by instantiating the central infrastructure. It resolves the variation points according to specific product requirements; and
- PL Management: includes activities to develop and maintain a PL. Strategies and methods are defined to manage the PL variabilities as, for instance, the SMarty approach [13] for UML-based variability management.

Feature models are used to capture and manage similarities and variabilities of a PL. A feature model encompasses mandatory, optional or alternative features. This model is created during the Domain Engineering activity as part of the central infrastructure. Then, it is used as input for the Application Engineering activity [5].

C. Application of the PL Approach to the UAV Domain

The application of the PL approach is important for developing UAVs as, according to specific application requirements, modifications of hardware or software architecture may occur. The PL approach can be used to decrease the UAV modification costs and effort, increasing the return on investment (ROI) [1].

The Simulink tool has several blocksets that can be used to model UAVs. Each blockset contains a group of blocks with specific functions for a given application domain. According to Dziobek et al. [7], such blocks can be used to represent PL variabilities.

Bottorweck et al. [3] present the concept of negative variability applied to Simulink models [9]. A negative variability represents the dependency relationships between domain variabilities and structural models, in this case Simulink models. It selectively filters out certain components based on a given configuration. In order to prepare for the Application Engineering activity, it is necessary to map features to the corresponding Simulink components earlier in the Domain Engineering.

Figure 2 presents the interaction between the application features and the blocks that represent such features in Simulink. In this figure, during the PL configuration activity, feature #2 and, consequently, block #3 and its relationships are eliminated, whereas feature #1, and block #4 are selected.

III. AN APPROACH FOR MAPPING PL FEATURES TO AN UAV MODEL

The mapping of PL features to UAV models, which is proposed in this paper, is supported by a process which consists of six steps that express a solution to represent negative variabilities. Such a process represents a solution to link PL features to UAV models by, for instance, allowing the interaction of Simulink and pure::variants tools. Figure 3 presents the process and its steps.

An application example based on the mini-UAV [4] is shown to illustrate each process phase. The variabilities are representations of the basic sensors with three GPS (Global Position System) alternative receivers, a mandatory accelerometer, and an optional gyroscope. Steps three and six are automatic, whereas remaining steps are supported by tools:

- Step 1 - Create initial feature model: an initial feature model is produced using pure::variants tool. This model represents the main functional and non-functional features of the system to be developed. Figure 4 presents an excerpt of the mini-UAV feature model focused on basic sensors. It is possible to select three different GPS receiver brands (A, B or C); and (ii) the gyroscope subsystem, which is an optional selection as it has an enabled block that takes as input the constant value VARGyro;
- Step 2 - Create Simulink model: the initial feature model created in Step 1, which represents a higher level of abstraction, is used to create the Simulink model. Figure 5 presents the mini-UAV sensors subsystem with the following variabilities represented: (i) the kind of sensor, which requires an alternative selection (A, B and/or C); and (ii) the gyroscope subsystem, which is an optional selection as it has an enabled block that takes as input the constant value VARGyro;
- Step 3 - Import Simulink model to Eclipse: imports Simulink basic blocks as a feature model to the Eclipse IDE [8], in a Pure::Variants predefined format. This step is automatically carried out by the SimulinkImport tool developed as an Eclipse plugin. The main features...
of the feature model are imported hierarchically to represent the mini-UAV subsystems, as shown in Figure 6:

- **Step 4 - Add relationships and constraints to initial feature model:** relationships and constraints are added to the initial feature model, created in Step 1, for mapping which feature is related to which Simulink element (e.g., a switch), thus generating the produced feature model. Such a mapping is described by conditional expressions, such as requires and conflicts and it can map one PL feature to one or more Simulink elements;

- **Step 5 - Configure produced feature model:** the produced feature model is configured by instantiating it as shown in Figure 7. It will allow the generation of a product with resolved variabilities; and

- **Step 6 - Generate pruned Simulink model:** the relationships and constraints between features and Simulink blocks are used to resolve the variabilities. Then, Simulink is able to generate the code of the new product. Figure 8 presents the Simulink blocks for the generated product, created from the instantiation of the mini-UAV model (Figure 5).

### IV. RELATED WORK

Several works in the literature are related to ours. However, they use aspects, DSLs (Domain-Specific Languages), and specific transformation models to handle PL variabilities and embedded systems models.

Dziobek et al. [7] developed the pure::variants Connector tool that allows the synchronized mapping between PL variabilities and Simulink functional variants. They also present strategies of how to apply the variability concept to Simulink models. However, such a tool is not free and it runs exclusively in Windows platform.

Polzer et al. [14] designed a PL for critical embedded systems that uses development techniques based on models and Rapid Control Prototyping (RCP). A case study presents the application of the negative variability concept to a robot. The pure::variants Connector tool is used to model variability.

Botterweck et al. [3] use DSLs and Model-to-Model (M2M) transformation techniques to create configurable
Figure 3. Proposed Process and its Steps for Mapping PL Features to UAV Models.

models with formal semantics. The approach is demonstrated for PLs of embedded systems using Simulink as a DSL for the model-based engineering of embedded systems.

Voelter and Groher [17] used aspect-oriented and model-driven techniques to implement PLs. Their approach is based on variability mechanisms demonstrated with a sample PL for home automation applications.

V. CONCLUSION AND FUTURE WORK

This paper presented concepts with regard to PL, UAVs and PL development for UAVs. Such concepts are essential
to the proposed approach that aims at supporting the mapping of PL features to UAV models. In order to make it possible, it was developed a plugin, the SimulinkImport, which is capable of importing Simulink models and their main subsystems to the Eclipse platform. Unlike pure::variants Connector, the SimulinkImport plugin is multiplatform and adds mapping keys to every subsystem in the imported Simulink code, allowing its instantiation.

SimulinkImport and pure::variants Connector can run in parallel since their objectives are quite different, although they use the same tool settings. Therefore, SimulinkImport relates PL features to Simulink model elements to create a specific product instance.

It was also presented an example, taking into account a mini-UAV, of the application of the proposed approach to demonstrate the mapping process, as well as the SimulinkImport generated artifacts. Currently, such a mapping process presents certain limitations as, for instance, the lack of: (i) standardization of UAV feature models, and
(ii) models to improve the management of the involved mapping elements.

Directions for future work and contributions include:

- apply the proposed approach to a real UAV;
- extend the proposed approach to encompass Model-driven Engineering (MDE) concepts to represent negative variabilities and transformation of models;
- improve the SimulinkImport tool to support traceability from feature model to lower-level PL UAV models;
- the investigation of UAV code generation by means of generative programming techniques; and
- plan and conduct controlled experiments to analyze the proposed approach’s feasibility by applying it to a real UAV PL in industry and taking well-qualified professionals as subjects.

ACKNOWLEDGEMENTS

The authors would like to thank National Institute of Science and Technology for Critical Embedded Systems (INCT-SEC/Brazil) for supporting this study.

REFERENCES


