# *SyMPLES* *A SysML-based Approach for Developing Embedded Systems Software Product Lines*

Rogério F. Silva[1], Vanderson H. Fragal[1], Edson A. Oliveira Junior[1], Itana M. S. Gimenes[1] and Flávio Oquendo[2]
[1]*Departamento de Informática, Universidade Estadual de Maringá, Maringá-PR, Brazil*
[2]*IRISA, European University of Brittany, UBS, Vannes Cedex, France*

Keywords: Embedded Systems, Software Product Line, SysML, Unmanned Aerial Vehicle, Variability Management.

Abstract: The evolution of hardware platforms has transferred a great amount of functionality to embedded software, thus increasing its complexity. The Software Product Line approach (SPL) has been successfully applied to the development of embedded software both to deal with complexity and to accelerate time to market. This paper contributes to enhance the application of SPL to embedded systems by extending the SysML language to include variability as well as by providing a well-defined SPL development process. The proposed approach, named SysML-based Product Line Approach for Embedded Systems (*SyMPLES*), includes two SysML extensions, created by means of the UML profiling mechanism both to express variability concepts and to associate SysML blocks to the main classes of functional blocks. An application example was developed for two subsystems of an Unmanned Aerial Vehicle (UAV) family, named Tiriba, which has been produced by the AGX Company in cooperation with the National Institute of Science and Technology for Safety-Critical Embedded Systems (INCT-SEC).

## 1 INTRODUCTION

Embedded systems are applications for processing embedded information in a larger product which is not usually directly visible to users (Marwedel, 2006). The increased computational power of hardware platforms has led to a fast growth of embedded software over the last decades; this is mainly due to the transfer of more functionality to software (Burch et al., 2001). As a consequence, embedded systems became larger and more complex, thus more demanding in terms of software engineering techniques (Bassi et al., 2011). The Software Product Line (SPL) approach has been successfully applied to embedded systems (Polzer et al., 2009); (Shimabukuro et al., 2011); (Achatz, 2011). However, there is a need to improve the specification of higher-level models of embedded systems with respect to variability representation. Such models are usually specified in UML (Farkas et al., 2009); (Moreira et al., 2010). This language does not take into account important aspects of the system engineering discipline, which involve the development of complex systems, implemented by hardware and software solutions.

This paper presents *SyMPLES*, a SPL approach that is based on higher-level models specified in SysML enhanced with extensions that represent variability concepts. SysML is a language designed for embedded systems, derived from UML and officially adopted and standardized by the OMG (OMG, 2012). SysML extends the concept UML classes using the concept of block to model not only software, but also hardware and any other constituent of a system. SysML offers important modelling features for embedded systems (Sabetzadeh et al., 2011), which include:

- better semantical expression of systems engineering features, thus reducing the bias of UML towards classical software. In particular, the block concept has been introduced, which is a modular unit of system description used to describe structural concepts in a system and its environment.
- improved requirement specification. This allows the definition of both function and non-

F. Silva R., H. Fragal V., A. Oliveira Junior E., M. S. Gimenes I. and Oquendo F..
SyMPLES - A SysML-based Approach for Developing Embedded Systems Software Product Lines.
DOI: 10.5220/0004446802570264
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 257-264
ISBN: 978-989-8565-60-0
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

functional requirement, in addition to their tracing to design models described at different levels of abstraction.

*SyMPLES* includes two SysML extensions for representing embedded system artifacts, created by means of the UML profiling mechanism: (i) *SyMPLES-ProfileVar* which supports variability representation by providing a set of stereotypes and tagged values; and (ii) *SyMPLES-ProfileFB* which includes a set of stereotypes that represents the main classes of functional blocks. These stereotypes represent the behavior associated with standard SysML blocks.

In addition to the SysML extensions, *SyMPLES* defines two processes: (i) *SyMPLES-ProcessPL* which defines a set of activities and guidelines to support the creation of the SPL artifacts; and (ii) *SyMPLES-ProfileVar* which is concurrently executed with the first process and contains a set of activities and guidelines to support the identification, delimitation and representation of variability, as well as the SPL product configuration.

An application example of *SyMPLES* was carried out to design a SPL for a family of UAV, called Tiriba, developed by the AGX Company (AGX, 2012), in partnership with INCT-SEC (INCT-SEC, 2008). Tiriba is used in pre-defined missions and applications such as agricultural and environmental monitoring. This family of aircrafts has taken advantage of the miniaturization of electronic components such as GPS receivers, digital cameras, wireless communication equipment and sensors (Branco *et al*., 2011) to reduce production costs and expand its application domain

This paper is organized as follows: Section II presents a background summary; Section III presents the *SyMPLES* approach; Section IV presents an application example of the *SyMPLES* approach to develop a SPL for the Tiriba UAV Family. Section V presents discussion and related works; and Section VI presents conclusions and ongoing works.

## 2 BACKGROUND

Important concepts related to the application of the SPL approach to system engineering are presented in this section.

### 2.1 SPL applied to the embedded Systems Domain

A SPL describes a family of systems that share a common and managed set of features according to the requirements of a particular market segment (Linden et al., 2007). The main SPL engineering activities are: (i) domain engineering in which a core infrastructure is designed by explicitly representing the variation points that allow further configuration; and (ii) the application engineering in which the configuration of the core components takes place to resolve variation points. Feature models are used for capturing and managing the similarities and variabilities of a SPL. A feature model may contain mandatory, optional and alternative features. Such a model is created during domain engineering as part of the core infrastructure, and then used as input to the application engineering (Czarnecki et al., 2005).

The SPL approach has been important for the embedded systems domain (Polzer et al., 2009); (Buschmann and Schwanninger, 2011); (Achatz, 2011) because the market usually produces similar product models of which the specific requirements usually lead to changes in the core software architecture of the family of products. The use of SPL can make the systematic evolution of products easier, thus reducing development cost, effort and time to market.

In addition to the SPL principles, we have used an approach to manage variability named *SMarty* (Oliveira Junior et al., 2010) as it allows the representation of variability in UML-like models and has also an explicit process for identification, delimitation and tracing variabilities.

### 2.2 Systems Engineering and SysML

Systems engineering is a multidisciplinary approach that aims to develop complex systems implemented using solutions that encompass hardware and/or software (Lykins and Friedenthal, 2000; Weilkiens, 2007). Embedded systems are within the scope of system engineering techniques. In this paper, we take embedded systems as the domain of our SPL therefore we conceived it by taking into account languages and models previously applied to system engineering.

SysML was used as the main notation for our SPL artifacts. This language (OMG, 2012) is quickly becoming a de-facto standard for systems engineering (Sabetzadeh et al., 2011). It reuses a subset of diagrams defined in UML (version 2.3) and defines its own extensions. The main diagrams used in our approach were the Block Definition Diagram and the Internal Block Diagram because they are related to functional block models usually applied by industry to specify embedded systems

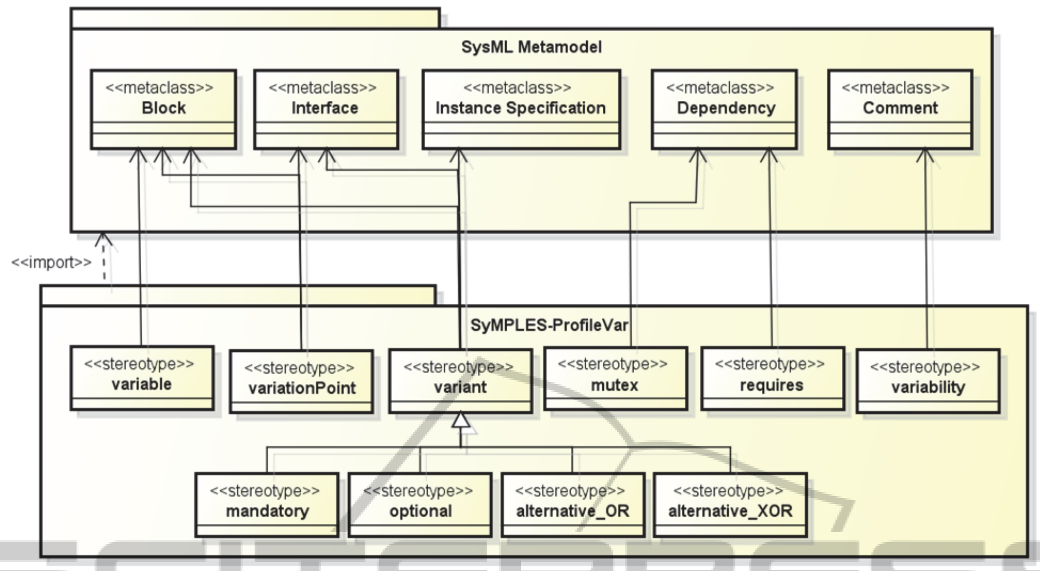

Figure 1: Stereotypes of the *SyMPLES-ProfileVar* profile.

(Sjostedt, 2008).

Model-based Systems Engineering (MBSE) is an approach that uses higher-level models represented in languages such as SysML. MBSE processes (INCOSE, 2006) encompass requirements specification, design, analysis, verification and validation of system design activities. Examples of MBSE processes are: Harmony (Douglass, 2004), RUP SE (Murray, 2003) and OOSEM (Lykins and Friedenthal, 2000). *SyMPLES* is mainly based on OOSEM as it uses SysML and is a tool-independent process. In addition, it has the advantage of being generic and providing a framework for instantiating methods for system engineering, as proposed in (Bassi *et al*., 2011).

## 3 THE *SyMPLES* APPROACH

*SyMPLES* provides two profiles and two processes. The profiles are extensions of the SysML language created to support the representations of the SPL artifacts and the processes guide the user in the application of the profiles to specify the SPL artifacts. The profiles and processes are described in the next sections.

### 3.1 *Sy1* Profiles

a) The *SyMPLES Profile for Functional Blocks* (*SyMPLES-ProfileFB*) provides additional semantics to SysML blocks. It is composed of a group of stereotypes to support the mapping of the SysML elements to main classes of functional blocks languages, such as Simulink. This supports the association of behavior with SysML models, thus facilitating the transformation process from specification to implementation; and,
b) The *SyMPLES Profile for Representation of Variability* (*SyMPLES-ProfileVar*) is based on the UML profile defined in the SMarty approach (Oliveira Junior *et al*., 2010). It defines a set of stereotypes and tagged values that allow the association of SysML elements such as Block, Interfaces, Dependency and Comment with variability concepts. It enables the specification of the structural, behavioral and variability aspects by using a single notation. Figure 1 presents the *SyMPLES-ProfileVar* and its stereotypes. We can see, for instance, that the `variable` stereotype is applied to the `Block` metaclass.

### 3.2 *SyMPLES* Processes

*SyMPLES* processes are composed of a set of activities as follows:

a) *The SyMPLES Process for Product Lines* (*SyMPLES*-ProcessPL) defines a set of generic and tool-independent activities that supports the SPL domain engineering; and,
b) *The SyMPLES Process for Identification of Variabilities (SyMPLES-ProcessVar)* is an iterative and incremental process for representing and managing SPL variability based on the

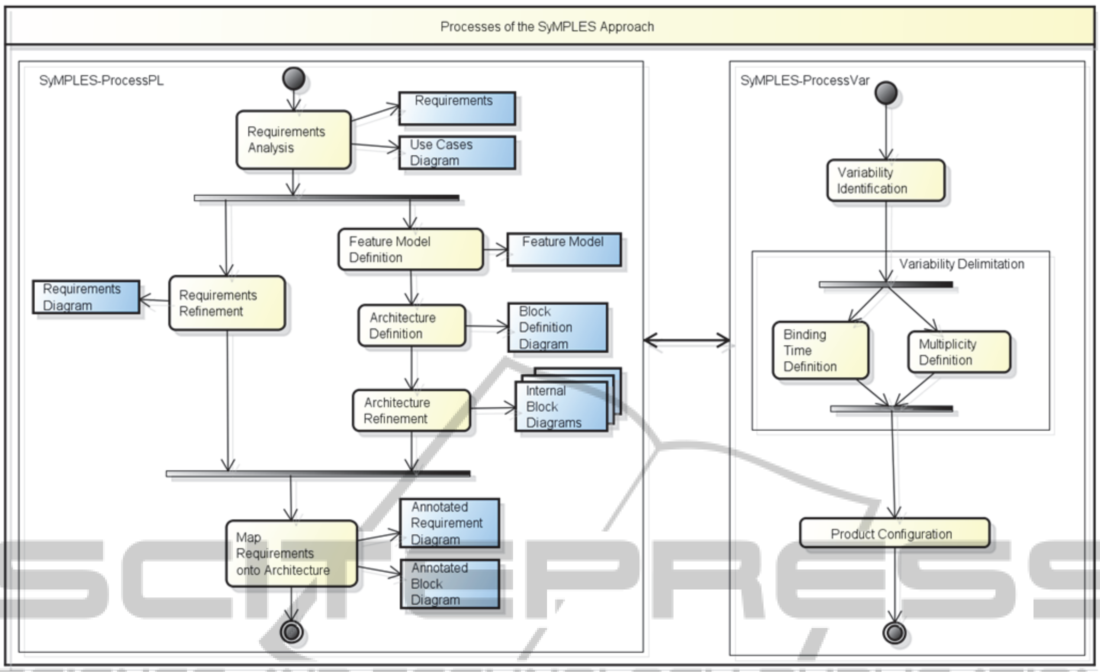

Figure 2: Interaction Between *SyMPLES*-ProcessPL and *SyMPLES*-ProcessVar.

SMarty approach (Oliveira Junior et al., 2010). Each *SyMPLES-ProcessPL* activity requires iteration with the *SyMPLES-ProcessVar*. Its goal is to support the user in the identification, delimitation, representation and configuration of variability.

Figure 2 shows an activity diagram which represents the interaction between the *SyMPLES-ProcessPL*, represented by the activities of the rectangle on the left side, and *SyMPLES-ProcessVar*, represented by the activities of the rectangle on the right side. These processes run in parallel during SPL domain engineering.

The activities of *SyMPLES-ProcessPL* were extended from OOSEM (Bassi et al., 2011), as follows:

- Requirements analysis, which performs analysis of both the system environment and user needs, for generating a list of requirements and the system use case diagram;
- Requirements refinement, which takes as input a list of requirements and the use case diagram produced in the previous activity, and generates as output a SysML requirements diagram; Feature model definition, which identifies the externally visible features of products that compose the SPL and organizes them into a feature model;
- Architecture definition, which decomposes the system into blocks to create the SysML block diagram, and therefore define how such blocks interact to meet the system requirements;
- Architecture refinement, which defines the internal structure of the blocks by creating SysML internal block diagrams; and
- Map requirements onto architecture, which associates previously created blocks and the requirements defined in the requirements diagram. This activity produces tracing reports between the requirements and the architecture which support the analysis of the SPL evolution. SysML use, at this level, the requirements diagram instead of the use case model as this diagram allows the representation of non-functional requirements and a better mapping of them to the elements of the architecture.

*SyMPLES-ProcessVar* activities are described as follow:

- Variability identification: performed in every interaction between *SyMPLES-ProcessPL* and *SyMPLES-ProcessVar* taking as input the feature model, use cases, requirements and blocks. This activity aims to identify the variabilities associated with these models. *SyMPLES-ProfileVar* supports this activity by applying their stereotypes to SysML models and associating values for tagged values;
- Variability delimitation: which sets values for the

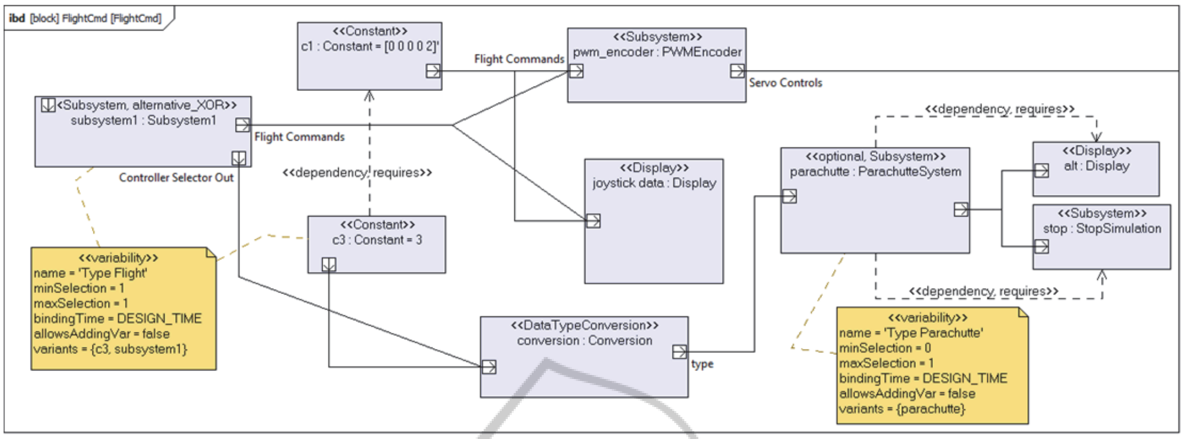

Figure 3: Excerpt of the Tiriba UAV flight control subsystems in SysML.

following variability tagged values: *minSelection*, *maxSelection*, *bindingTime*, *allowsAddingVar* and *variants*;

- Product configuration: which contains specific block models (eg. block and block definition diagrams). This activity leads to the resolution of SysML model variabilities for generating specification for a specific product. The product configuration can be performed manually or automatically as, for instance, using the tool pure::variants (Pure-Systems, 2011).

In addition to the activities described above, *SyMPLES* provides a set of guidelines to support the developer on how to develop and evolve the models.

## 4 APPLYING *SyMPLES* TO DEVELOP A SPL FOR UAV

### 4.1 Modeling the SPL for UAV

The activities defined by *SyMPLES-ProcessPL* were executed to create the core of a SPL for the navigation and flight control subsystems of the Tiriba UAV family. After the execution of the process activities, the following artifacts were produced: use case model, requirements diagram, feature model, block definition diagram and internal block diagram. We have recovered the specification of the Tiriba UAV block models from their existing Simulink models. Thus, the stereotypes of *SyMPLES-ProfileFB* were used to represent the main blocks recovered from the Simulink models in the SysML models.

The block definition and internal block diagrams were specified with their respective variabilities following the *SyMPLES-ProfileVar*. Figure 3 shows an excerpt of the internal block diagram of the UAV flight control subsystem. In this figure, blocks `c1` and `c3` were marked with the stereotype <<Constant>> indicating that they correspond to a block of type constant in Simulink. This shows that *SyMPLES-ProfileFB* supports the establishment of straightforward correspondence between SysML blocks and Simulink implementation blocks.

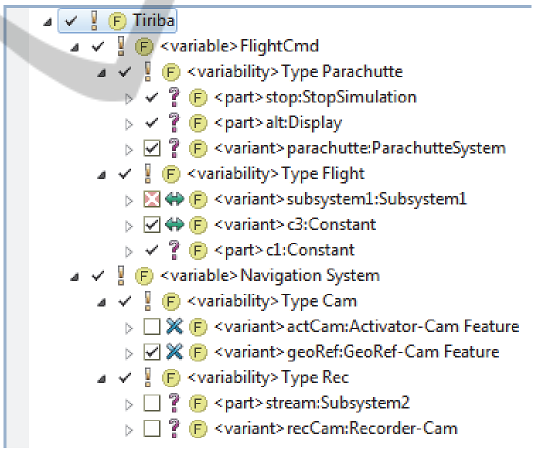
- Tiriba
  - <variable> FlightCmd
    - <variability> Type Parachutte
      - <part> stop:StopSimulation
      - <part> alt:Display
      - <variant> parachutte:ParachutteSystem
    - <variability> Type Flight
      - <variant> subsystem1:Subsystem1
      - <variant> c3:Constant
      - <part> c1:Constant
  - <variable> Navigation System
    - <variability> Type Cam
      - <variant> actCam:Activator-Cam Feature
      - <variant> geoRef:GeoRef-Cam Feature
    - <variability> Type Rec
      - <part> stream:Subsystem2
      - <variant> recCam:Recorder-Cam

Figure 4: Tiriba UAV Feature Model in pure::variants.

After executing the variability identification activity of *SyMPLES-ProcessVar*, the blocks `subsystem1` and `c3`, shown in Figure 3 were identified as exclusive variants, thus they were annotated with <<alternative_XOR>> stereotype. This means that only one of these blocks is selected after the product configuration. The block `parachute` was identified as an inclusive variant; this means that the block may be selected to a specific product.

The product configuration activity of *SyMPLES* was suported by an Eclipse plugin, named

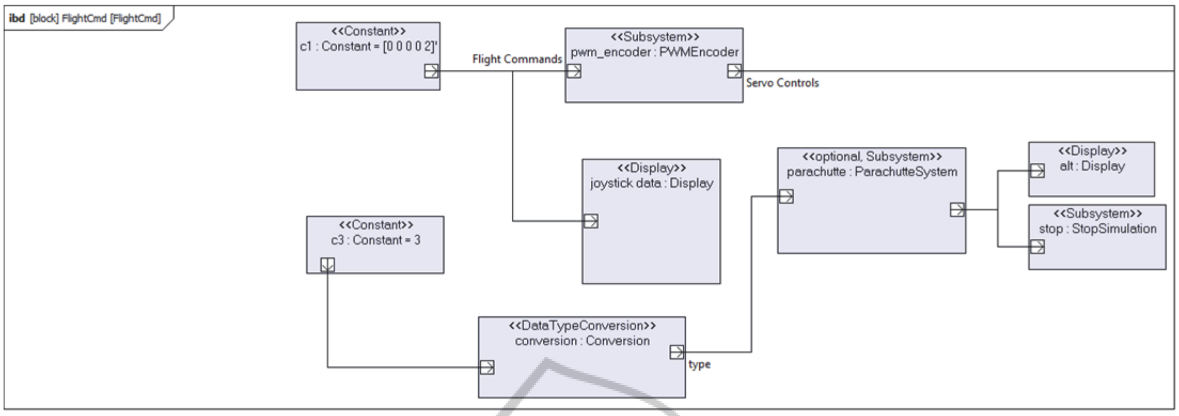

Figure 5: Excerpt of the flight control subsystem of the Tiriba UAV after the variability configuration.

*SysMLImporter*. It was implemented to map SysML block models with variability to feature models in the pure::variants tool, as shown in Figure 4. The configuration of the feature model results in SysML models for specific products.

Important elements presented in the Figure 4 are:

- Variation points are identified with <variability>; in this example they are `Type Parachute`, `Type Flight`, `Type Cam` e `Type Rec`; the variation points `Type Flight` and `Type Parachute` are parts of the navigation system and can also be seen in Figure 3;
- Variants associated with the variation points are identified with <variant>;
- <part> identifies elements of the SysML model which are necessary after the selection of a certain variant; for instance, in Figure 3 the blocks `alt` and `stop` are automatically selected after the selection of the `parachute block`, because they are linked by a <<requires>> relationship.

After selecting the features of the specific product the *SysML Importer* plugin can clean up the correspondent SysML model eliminating the features that were not selected. Figure 5 shows the SysML model of the flight control subsystem of the Tiriba UAV obtained from the configuration selected in Figure 4. In this figure, the blocks `c1` and `c3` are not present because in the alternative exclusive variability `Type Flight` of the feature model (Figure 4) the block `c3` was selected and it requires block `c1`. The block `parachute` is also in the model because the alternative variant was selected.

## 5 DISCUSSION AND RELATED WORK

*SyMPLES* application was analyzed according to main aspects as follows:

- <u>Use of High-level Abstraction Models and the SysML Language</u>: *SyMPLES* uses SysML for modeling embedded systems. This language proved to offer appropriate resources to represent different perspectives of the system architecture through requirements, blocks, ports, parametric diagrams and allocations. It is also possible to trace the model evolution throughout the development process. In *SyMPLES* we could observe the transformation of requirements to the system block diagram. Perseil and Pautet (2010) and Zaki and Jawawi (2011) use the MARTE profile (OMG, 2009) for modeling embedded systems. However, MARTE focuses on the specification of real-time issues, which involve guidelines to annotate models based on time, such as performance and scalability. Farkas et al., (2009) propose an approach that uses UML to specify embedded systems integrated with functional blocks specified in Simulink. However, as stated previously, UML is a general-purpose language and it does not allow the specification of details required in system engineering such as non-functional requirements. So, by working with SysML we provided an approach closer to what system engineering requires.
- <u>Application of SPL Techniques:</u> it was possible to represent variabilities in SysML models based on the *SyMPLES-ProfileVar* profile, which takes SMarty as a basis for proposing new extensions for the SysML language. SMarty provides a

more precise representation of variability issues by making explicit relationships between variation points and variants. Thus, *SyMPLES-ProfileVar* inherits such characteristics form SMarty. The *SyMPLES-ProfileFB* also makes it easier to associate the SysML blocks to functional block languages, such as Simulink. In addition, the processes *SyMPLES-ProcessPL* and *SyMPLES-ProcessVar* have well-defined activities and guidelines that make the SPL development easier. Botterweck et al., (2009) propose an approach to develop SPL for embedded systems in which Simulink models are translated to feature models. Thus, it is possible to associate Simulink blocks with features in order to obtain specific models from the configuration and their correspondent implementation.

- Support for Product Configuration: the *SyMPLES-ProcessVar* established a set of guidelines that made product configuration systematic. This made it possible to implement *SysMLImporter*, a tool that maps SysML models to pure::variants feature models that generates specific SysML models for the products.

## 6 CONCLUSIONS AND FUTURE WORK

This paper presents *SyMPLES*, an approach to develop SPL for embedded systems. It was applied to model two subsystems of a real UAV application. The SysML models presented were created in a reverse engineering process, using the real Simulink models of the UAV Tiriba. In addition, the activities defined in the *SyMPLES-Process*PL includes requirements analysis and makes it possible to create SysML models using the stereotypes for functional blocks defined in the SyMPLES-ProfileFB, in a straightforward development task.

One of the key contributions of our work is to provide high-level abstraction models in SysML enhanced with a variability management mechanism. *SyMPLES* is based on SysML, because this language enable the expression of important system engineering concepts which cannot be represented in the general UML, such as: parametrized blocks, hierarchically structured requirements, mapping between requirements and the system architecture and the definition of ports and flows.

In addition, *SyMPLES* provides support to:

- Define the semantics of functional blocks of SysML through the *SyMPLES-ProfileFB*; as can be seen in Figures 3 and 5, the application of the stereotypes of the profile *SyMPLES-ProfileFB* to the block model of the Tiriba UAV indicate the correspondent block in *Simulink*. By specifying the system models initially at a higher abstraction level, make it possible to improve tasks like requirements specification, software architecture analysis and model-based testing. The use of the profile *SyMPLES-ProfileVar* enables the representation of variability concepts directly in SysML without adding artificial elements to represent variability.
- Provide two well-defined processes to guide the user in the specification of the SPL artifacts in SysML by identifying, delimiting and representing variability;
- Offer a mechanism to support variability configuration. This mechanism was implemented as an Eclipse plugin to associate SysML models with feature models in pure::variants.

Ongoing work includes the completion of the SPL mechanisms necessary in the application engineering. This uses model transformation techniques based on the MDA approach (OMG, 2004) to enable the generation of Simulink models from the SysML specification of a product generated by *SyMPLES*.

## ACKNOWLEDGEMENTS

We are grateful to the Brazilian funding agencies CNPq/INCT-SEC and FAPESP for supporting this work.

## REFERENCES

Achatz, R. Product Line Engineering at Siemens -- Challenges and Success Factors: A Report on Industrial Experiences in Product Line Engineering. In: *Proc. of 15th International Software Product Line Conference (SPLC 2011)*, Munich, 2011, p. 10-11.

AGX Tecnologia Ltda, www.agx.com.br, accessed in 12/19/2012.

Bassi, L., Secchi, C., Bonfé, M., and Fantuzzi, C. A SysML-Based Methodology for Manufacturing Machinery Modeling and Design. In: *IEEE/ASME Transactions on Mechatronics*, v. 16, n. 6, p. 1049-1062, 2011.

Botterweck, G., Polzer, A., and Kowalewski, S. Interactive Configuration of Embedded Systems

Product Lines. In *Proc. of the International Workshop on Model-Driven Approaches in Product Line Engineering*, San Francisco, 2009, p. 51-57.

Branco, K. R. L. J. C.; Pelizzoni, J. M.; Oliveira Neris, L.; Trindade, O.; Osorio, F. S.; Wolf, D. F. Tiriba - a new approach of UAV based on model driven development and multiprocessors. In: *IEEE International Conference on Robotics and Automation (ICRA 2011)*, Shanghai, 2011, p. 1-4.

Burch, J., Passerone, R., and Sangiovanni-Vicentelli, A. L. Using Multiple Levels of Abstractions in Embedded Software Design. In: *Proc. of the International Workshop on Embedded Software (EMSOFT 2001),* Berlin, 2001, p. 324-343.

Buschmann, F.; Schwanninger, C. Successful Product Line Engineering: Experiences from the Real World. *Proc. of 15th International Software Product Line Conference (SPLC 2011),* Munich, 2011, p. 349.

Czarnecki, K., Helsen, S., and Eisenecker, U. Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models. *Software Process Improvement and Practice*, v.10, n. 2, p. 143-169, 2005.

Douglass, B. P. *Real-Time UML: Developing Efficient Objects for Embedded Systems*. 3rd ed. Reading, MA: Addison-Wesley, 752 p. 2004.

Farkas, T., Meiseki, E., Neumann, C., Okano, K., Hinnerichs, A., and Kamiya, S. Integration of UML with Simulink into Embedded Software Engineering. In: *Proc. of the ICROS-SICE International Joint Conference*, Fukuoka, Japan, 2009, p. 474-479.

INCOSE. *International Council on Systems Engineering*. *Object Oriented System Engineering Method*. OOSEM Descriptive Outline for INCOSE SE Handbook Version 3, Annotated Update, Sect. 6.4.2, p. 6-1 a 6-6, 2006.

INCT-SEC, *Sistemas Embarcados Críticos: aplicações em segurança e agricultura*, CNPq 2008, Available in: <http://www.inct-sec.org>. Accessed in: Sep. 18, 2012.

Linden, F., Schmif, K., and Rommes, E. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Secaucus: Springer, 2007, 353 p.

Lykins, F. M., and Friedenthal, S. Adapting UML for an Object-Oriented Systems Engineering Method (OOSEM). In: *Proc. of the INCOSE International Symposium*, Minneapolis, 2000, 91-98.

Marwedel, P. *Embedded System Design*. Springer, Dortmund, 2006, 241 p.

Moreira, T. G., Wehrmeister, M. A., Pereira, C. E., Pétin, J., and Levrat, E. Automatic Code Generation for Embedded Systems: From UML Specifications to VHDL Code. In: *Proc. of the 8th IEEE International Conference on Industrial Informatics*, Osaka, Japan, 2010, p. 1085-1090.

Murray, C. *RUP SE:* The Rational Unified Process for Systems Engineering. The Rational Edge, *Rational Software*, 2003.

Oliveira Junior, E. A., Gimenes, I. M. S., and Maldonado, J. C. Systematic Management of Variability in UML-based Software Product Lines, *Journal of Universal Computer Science*, v.16, p. 2374-2393, 2010.

OMG. *Model-Driven Architecture*. Object Management Group. 2004.

OMG. *MARTE UML Profile Specification*. Object Management Group. 2009.

OMG. *Systems Modeling Language (SysML)*, 2012. Version 1.3. Object Management Group.

Perseil, I., and Pautet, L. High-Level Abstraction Modeling for Detailed Analysis of Avionic Real-time Systems. In *Proc. of the 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, 2010, p. 418-424.

Polzer, A., Kowalewski, S., and Botterweck, G. Applying Software Product Line Techniques in Model-based Embedded Systems Engineering. In: *Proc. of the 6th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, Vancouver, CA, 2009, p. 2-10.

Pure-Systems. pure::variants. Available in: <http://www.pure-systems.com/pure_variants.49.0.html>. Accessed in: Dec. 10, 2011.

Sabetzadeh, M.; Nejati, S.; Briand, L.; Mills, A. E. Using SysML for Modeling of Safety-Critical Software-Hardware Interfaces: Guidelines and Industry Experience. In *Proc. of the IEEE 13th International Symposium on High-Assurance Systems Engineering (HASE),* 2011, p. 193-201.

Shimabukuro, J.; Ohara, T.; Okamoto, C.; Atarashi, Y.; Koizumi, S.; Watanabe, S.; Funakoshi, K. *Key* Activities for Introducing Software Product Lines into Multiple Divisions: Experience at Hitachi. In: *Proc. of 15th International Software Product Line Conference (SPLC 2011)*, Munich, 2011, p. 261-266.

Sjostedt, C., Shi, J., Torngren, M., and Servat, D. Mapping Simulink to UML in the design of embedded systems: Investigating scenarios and transformations. In: *Proc. of the 4th Workshop on Object-oriented Modeling of Embedded Real-Time Systems*, Paderborn, Germany, 2008, p. 36-43.

Weilkiens, T. *Systems Engineering with SysML/UML*. Morgan Kaufmann Publishers. 2007, 320 p.

Zaki, M. Z. M., and Jawawi, D. N. A. Model-Based Methodology for Implementing MARTE in Embedded Real-Time Software. In *Proc. of the IEEE Symposium on Computers & Informatics*, 2011, p. 536-541.