Efficient Software Controller Variant Development and Validation (ECoVaDeVa) Overview of a Flemish ICON Project

Bart Meyers[1], Simon Van Mierlo[2], Davy Maes[1], and Hans Vangheluwe[2,3]

[1] Flanders Make vzw, Belgium
{bart.meyers, davy.maes}@flandersmake.be
[2] University of Antwerp - Flanders Make vzw, Belgium
{simon.vanmierlo, hans.vangheluwe}@uantwerpen.be
[3] McGill Unviversity, Canada

Abstract. This paper describes the goals, (partial) results and lessons learned of the ECoVaDeVa project, a Flemish project that groups academic and industrial partners around the efficient, model-based development of software controller variants for Cyber-Physical Systems (CPSs). ECoVaDeVa's high-level goal is to apply Product Line Engineering (PLE) techniques to CPS controller design in all phases of the development lifecycle (design, simulation, testing, deployment) as an extension of existing software product line techniques. While PLE is well researched in software development, it is not clear whether these results apply to CPS controller design. The added complexity stems from the heterogeneity of models representing the system, involving plant, controller and environment, software and hardware, and virtual test benches (model-in-the-loop, hardware-in-the-loop, etc.). The envisioned result of the project is a set of tools, techniques, and guidelines for the efficient management of CPS controller product variants. The techniques developed during the project are demonstrated on a common use case: a windshield wiper.

1 Project Details

Name and Acronym of the Project: Efficient Software Controller Variant Development and Validation (ECoVaDeVa)
List of Participants:
- AnSyMo/CoSyS-lab (University of Antwerp)
- CodesignS (Flanders Make Strategic Research Center)
- Dana Belgium
- Atlas Copco
- Siemens PLM Software Belgium

Link to Official Website:
https://www.flandersmake.be/en/projects/ecovadeva
Status and Duration: ongoing. Project start date: January 1st, 2017. Planned end date: June 30th, 2019.

2 Project Description

2.1 ICON Project Description

ECoVaDeVa is a Interdisciplinary Collaborative Research (ICON) project. An ICON project involves a strategic research center (including academic partners) and industrial partners. The goal is to transfer basic research to the industrial partners. In the project, a common research challenge is identified for all of the industrial partners. In the *strategic basic research* part, generic research questions are formulated and addressed to improve upon the state-of-the-art by the strategic research center. In the *applied research* part, the results of this research are then translated to the specific case of each of the industrial partners in separate work packages. This requires a close collaboration between academic and industry partners. In this paper, we focus on the results of the strategic basic research, as applied research results are confidential.

2.2 Problem Description and Goals

In a previous research project (VARIES[4]), Flanders Make (formerly FMTC) and Dana (formerly Spicer) investigated variability tools with respect to their applicability in real-life mechatronic applications beyond the classical toy examples. From that project, a number of research challenges were identified.

The main goal of this project is to increase the efficiency of the development and validation process of Cyber-Physical System (CPS) controller software variants. This efficiency gain is achieved by bridging the gap from business software Product Line Engineering (PLE) methods and tool prototypes only demonstrated on toy problems to industrial-scale CPS software development and validation processes. More specifically, the project aims to:

- Create a methodology and provide tool support for building central variability models for CPSs with variability in hardware and software architectures and automatically generating a configuration tool from these models;
- Provide insights to companies in the various possibilities to implement variability in behaviour simulation languages such as The MathWorks Simulink and SISW Imagine.Lab by offering them a decision tree for selecting the most appropriate variability mechanisms;
- Create a methodology and a toolbox that companies can use to automate their specific build and validation process of CPS software variants, including the generation of Model-in-the-Loop and Hardware-in-the-Loop tests;
- Create a consistency tool, that allows for early detection of inconsistencies between central variability models and controller and plant simulation tools if the product line evolves due to change requests.

2.3 Methodology Overview

Figure 1 illustrates the simplified idea of applying PLE techniques to CPS controller software variant development and validation. A key element of the ac-

[4] https://artemis-ia.eu/news/varies.html

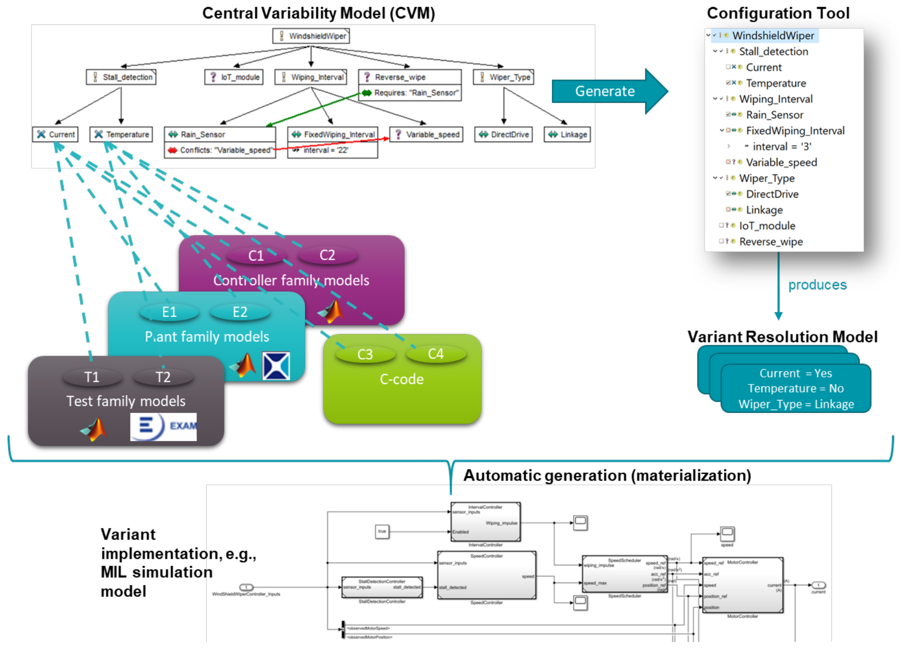

Fig. 1. Overview of the ECoVaDeVa approach.

cepted PLE approach [5] is the development of a (orthogonal) Central Variability Model (CVM) [2]. This model, often represented as a feature diagram [4], describes the set of all possible functionalities and parameters with which a product variant can be equipped and their mutual (in)compatibility. For instance, for a hydrostatic continuous variable transmission, one cannot select a chosen number of forward gears. Coupling this CVM to the controller, plant and test family model allows for the automatic generation of the controller software executable, the corresponding test suites, and the necessary plant models for performing the various XiL tests. To apply PLE techniques to CPS controller software variant development and validation, several aspects of Figure 1 require further research and will be investigated within the project:

- **[RQ1]** How to create/organize the CVM for real-life CPS controller software variants and to generate a configuration tool for the application engineer (top-level of Figure 1)?
- **[RQ2]** How to express variability in the modelling languages used by the various involved disciplines (bottom-left corner of Figure 1)?
- **[RQ3]** How to implement the actual build and validation process, taking into account CPS software specific practices such as heterogeneous tool chains, and X-in-the-loop (XiL) testing (we focus on model-in-the-loop (MiL) and hardware-in-the-loop (HiL) testing)?
- **[RQ4]** How to keep the various models consistent with each other if the product line is subject to a change request?

In order to validate and communicate new approaches within the basic research of the project, we established a windshield wiper controller use case. The case consists of CVMs and configuration tools, family models in Simulink, C++, MagicDraw, etc., and a fully automatic way to generate variants, covering all aspects of Figure 1. In the applied research of the project, the approaches that are investigated in the basic research are validated by the industrial partners in industry-scale environments.

3 Project Results

Selected results of the strategic basic research are explained in this section. During the project, we made use of the commercial tool pure::variants[5] to model the CVM and its relationship with modelling languages (RQ1 and RQ2).

Variability, Binding Times and Variant Generation. Approaches exist for implementing variability by means of variation points in modelling languages, but often, these are partial solutions. In the context of RQ2 and RQ3, it was investigated how existing constructs can be used to express variation points in relevant family modelling languages (Simulink, Amesim, SysML, etc.), and how they can be linked to the CVM. Different types of variation points (optional, alternative, multiple instances, etc.), different binding times (i.e., the moment in the design work flow a variant choice is applied, for example, at compile-time), and different levels of granularity (depending on the family modelling languages, this can be topological, connection, element property, etc.) are investigated. This includes the support for generating variants. A gap analysis for each used family modelling language discovered missing constructs. A notable example is the lack of support for model-time variability in Simulink, where Simulink model variants are automatically generated from a Simulink family model. A remedy was achieved by using rule-based model transformation for Simulink [3]. The transformation rules were defined using Simulink.

Consistency. In current approaches, it may be possible that variants are generated from a product family, that result in errors when building/simulating them. This is caused by a mistake in the family model, but such errors are notoriously hard to find in a product family, because of the interdependencies of variation points within the family model. This hinders the short and predictable product delivery times, aimed for by automatic generation of variants. In the context of RQ4 and based on an approach for UML [1], we have developed an approach to detect inconsistencies that result in build/simulation errors early. These are inconsistencies between CVM and family model, and are detected at the level of the product family instead of the level of the variant. Errors can thus be detected and resolved before customer orders of product variants are requested, avoiding unexpectedly long delivery times. In its current form, our approach is compatible with Simulink, but the principles can be reused for other tools as well. In its current status, the approach is partly implemented as a software tool that automatically checks for errors and reports them to the user.

[5] http://www.pure-systems.com/products/pure-variants-9.html

Plant Variability. For plant modelling, acausal modelling languages provide more natural abstractions, as the physical world is acausal by nature. It is then appropriate to model the variation points, if multiple product variants exist, in such an acausal language as well. In the specific case of Simscape for example, variability mechanisms of Simulink [8] can be reused. Currently however, no dedicated variation point support exist for acausal modelling languages. In the context of RQ2, we investigated how variability can be expressed in three acausal modelling languages: Modelica, Simscape, and Amesim. We make the distinction between energy-conserving (domain-specific) networks that represent a set of mathematical equations, and physical signals, coming from a sensor in the plant. We showed that each of these languages has concepts to model variability, but certain types of variability cannot be expressed. As a special point of focus, we showed that some variability concepts can generate variants whose causality is different, which might be an unwanted side effect and needs to be taken into account.

Architectural Variability. In order to deal with controller software on an industrial scale, an architectural overview needs to be maintained. Partial solutions exist for e.g., the automotive sector, for which AUTOSAR has support for variability [7]. In this project, a solution that applies to CPSs in general is of interest. Notably, the SYSMOD Model-Based Systems Engineering toolbox [9] provides support for variability, by implementing each aspect of PLE as shown in Figure 1 (including CVM) as a SysML profile, but the approach provides very limited support for specifying variation points. Related to RQ1 and RQ2, we investigated variability modelling in SysML, including what components exist in the system and their interface and relationships, a link to implementation models and code, link to middleware, and deployment. The outcome of this research is that technically, SysML can be used together with a CVM to express variability, but improvements w.r.t. tool and language support for usability and readability are recommended.

Dissemination. We have built a physical demo of the windshield wiper use case to attract attention to the problem of variability management in CPS design. It has been exposed at the Flanders Make Symposium, the Hannover Messe and will be exposed at the MATLAB Expo 2019 Benelux.

4 Identified Research Challenges

Cyber-physical controller design typically follows a (derivative of a) V-model process, in order to deal with different levels (e.g., concept, system, component, implementation). Each layer comes with different models and tools. Some CVM features are associated to models at the concept level (e.g., use an automatic or manual transmission), whereas other features are only relevant further downstream (e.g., which clutch control algorithm will be used). The clutch algorithm feature is not yet available, and it may be unknown during concept design that this involves a variation point. Consequently, the CVM should be split up accordingly so that the right stakeholders have access to the right (partial) CVM at the right time in the development process. Additionally, variation points need to be modelled in the right model, and need to be bound at the desired moment (i.e., binding time) in the tool chain, conforming the layer in which they are defined. Another layer of features depends on the different tasks performed during the development process: a model from which software is generated may need to change if it is used for MiL testing. Approaches for multi-view variability exist (e.g., [6]). *However, existing approaches do not directly translate to the abovementioned V-model process, binding times and testing strategies times within the multi-discipline setting of CPS design.*

5 Conclusion

In the ECoVaDeVa project, academic and industrial researchers have investigated the applicability of variability techniques in CPS development. Notably, specific answers were provided for dealing with variability in a multi-discipline setting, consistency at family level, variability in plant modelling and variability for architectural models. Additionally, research challenges were identified related to managing variability in layered CPS design processes based on the V-model. The project is aligned with the core subject of STAF, as it aims to define methods, techniques and tools that deal with variability through model-driven engineering and model transformation.

References

1. M. Alférez, R. E. Lopez-Herrejon, A. Moreira, V. Amaral, and A. Egyed. Consistency checking in early software product line specifications - the VCC approach. *J. UCS*, 20(5):640–665, 2014.
2. S. Bühne, K. Lauenroth, and K. Pohl. Why is it not sufficient to model requirements variability with feature models? In *AURE'04*, pages 5–12, 2004.
3. J. Denil, P. J. Mosterman, and H. Vangheluwe. Rule-based model transformation for, and in simulink. In *SpringSim '14*, page 4. ACM, 2014.
4. K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute, November 1990.
5. K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering - Foundations, Principles, and Techniques.* Springer, 2005.
6. D. Rabiser, H. Prähofer, P. Grünbacher, M. Petruzelka, K. Eder, F. Angerer, M. Kromoser, and A. Grimmer. Multi-purpose, multi-level feature modeling of large-scale industrial software systems. *Software and System Modeling*, 17(3):913–938, 2018.
7. J. Thomas, C. Dziobek, and B. Hedenetz. Variability management in the autosar-based development of applications for in-vehicle systems. In *VaMoS'11*, pages 137–140, 2011.
8. J. Weiland and P. Manhart. A classification of modeling variability in simulink. In *VaMoS'14*, pages 7:1–7:8, 2014.
9. T. Weilkiens. *SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML, 2nd edition.* Tim Weilkiens, 12 2016.