D5.2a Description of pilots for enhancing computer-aided design for low-code development by applying synthesis

Author(s)	Wytse Oortwijn (TNO), Dennis Hendriks (TNO)
Version	v 1.0
Date	13/06/2023
Confidentiality	Public

1. Abstract

This deliverable describes two pilots performed by TNO-ESI together with Cordis and Additive Industries, for enhancing computer-aided design for low-code development, by applying synthesis. Specifying low-code (Cordis SUITE) models in a way to guarantee necessary safety/user requirements is difficult. We demonstrate that correct control conditions for low-code models can automatically be synthesized from requirements. These control conditions can then be added to the models to make them correct with respect to the requirements. Our first pilot demonstrates the benefits of combining synthesis with low-code development, and our second pilot demonstrates that this combination can indeed add value in a real-world industrial context.

2. Change log

Version, Date	Description
v 1.0, 13/06/2023	Initial draft

3. Table of Contents

4. Introduction

Low-code development of control software, like done with for instance Cordis SUITE[1], involves specifying software in terms of models, from which PLC code is generated. These models consist of only a small amount of code, e.g., conditions to go from one model state to another. The advantage is that developers do not have to write low-level PLC code, but rather specify from a higher level what the software system should do, in terms of models. This may for example ease communication between (multi-disciplinary) teams. On the other hand, being able to specify software in terms of models instead of code does not take away the challenges in developing software systems that are correct. It may still be very difficult to specify models that satisfy necessary safety/user requirements.

TNO-ESI[2] together with Cordis[1] and Additive Industries[3] have performed two pilots to investigate if, and how, low-code development can benefit from (parts of) synthesis-based engineering (SBE[4]). SBE is an engineering methodology that essentially combines models-driven engineering with computer-aided design, to synthesize controllers for the specified models that are correct by construction. Traditionally, the to-be-controlled system is modeled in a particular way that differs from the way the controller is modeled in low-code modeling software like Cordis SUITE. It is therefore not immediately evident that SBE can be applied in this context, whether it can be combined with the low-code way of working, with the hierarchical UML-like modeling approach, and whether it will then bring the same benefits as in other contexts. Ideally, the combination of low-code and SBE gives design assistance, and may help developers to create correct models, especially in situations where correctness is intricate and difficult to achieve.

In the two pilots that we performed, we focused in particular on requirements that are easy to write down, but often not so trivial to implement and guarantee. For such requirements we managed to automatically generate correct control conditions, which can then be added to Cordis SUITE models to make them correct with respect to the requirement.

An alternative to synthesizing correct control conditions from requirements, is applying verification techniques to check whether models satisfy their requirements, e.g., using mCRL2 (Stramaglia & Keiren, 2022). Although there are connections between verification and synthesis, their aim is slightly different. Verification involves iteratively checking whether a model satisfies particular requirements, and manually adapting the model if it does not satisfy one of these requirements. Synthesis on the other hand calculates in one go a so-called supervisor model from the requirements. The supervisor model contains extra control conditions that ensure that all specified requirements are satisfied in all states of the model. In that sense, SBE might be considered a constructive approach to designing correct software systems. Or in simpler words: while formal verification can point you to the problems in your model, synthesis can both find and solve these problems for you.

5. First pilot: guaranteeing a FIFO requirement using synthesis

The first pilot, performed by TNO-ESI and Cordis, considers a simple pick-and-place system, consisting of a number of locations that can hold products, and a picking mechanism that can pick a product from one location and place it onto another, given that the destination location is not yet occupied.

[1] See https://cordis-suite.com.
[2] See https://esi.nl.
[3] See https://additiveindustries.com.
[4] See https://eclipse.dev/escet/cif/synthesis-based-engineering.

Figure 1 visualizes this pick-and-place system, where the rectangles are locations (e.g., INOUT, DISP, etc.) and the arrows between locations indicate how products can move (for example, if the location INOUT holds a product and DISP is unoccupied, the product on INOUT may move to DISP by performing an action called 'start'). Locations can hold at most one product. The INOUT location is a bit special, since it only allows products that entered the system (via an 'enter' action) to take the 'start' action, and only products that are 'finished' are allowed to leave the system via the 'exit' action. Also CHK is special since it has a sensor to check products, and enables either the 'leave' or the 'redo' action depending on whether the product has been approved or not, respectively, by the sensor.

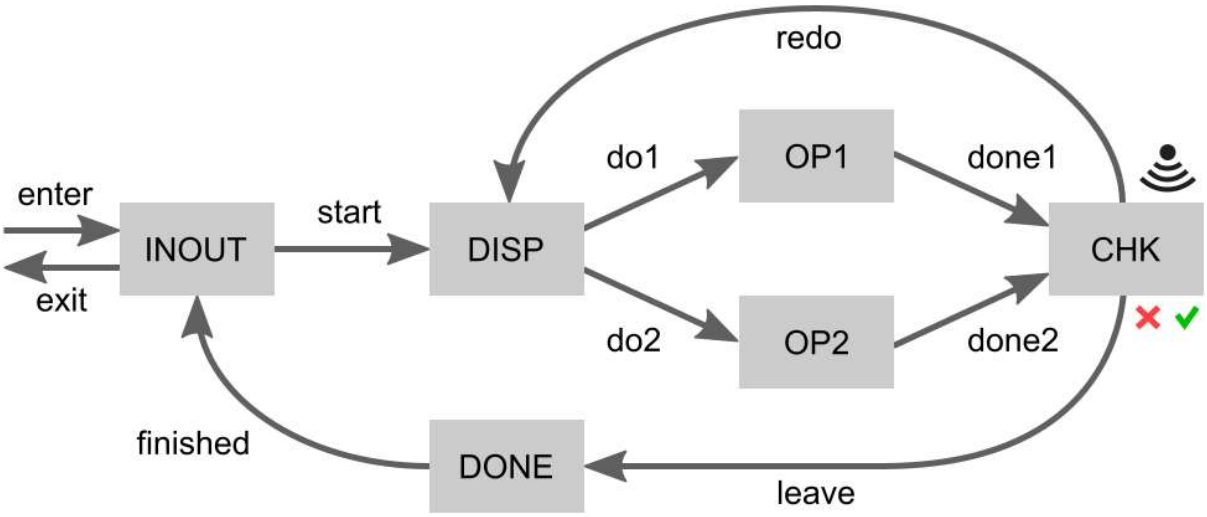

Figure 1: The simple pick-and-place system that we considered in the first pilot. Source: https://www.eclipse.org/escet/cif/synthesis-based-engineering/example.html, Accessed 2022-12-16.

Even though this pick-and-place system, by itself, is not difficult, it may be challenging to implement this system in such a way to satisfy certain requirements. For example, suppose that we want the system to adhere to a FIFO requirement: products leave the system in the same order as they enter the system. Then quite intricate restrictions on how products can move through the system need to be considered, as well as when products are allowed to enter the system. For example, one must be careful that the 'redo' action does not let products overtake each other, violating the FIFO requirement. Moreover, once must also be careful not to allow the system to deadlock, which may might happen if INOUT, DISP, OP1, CHK and DONE are all occupied, blocking any progress.

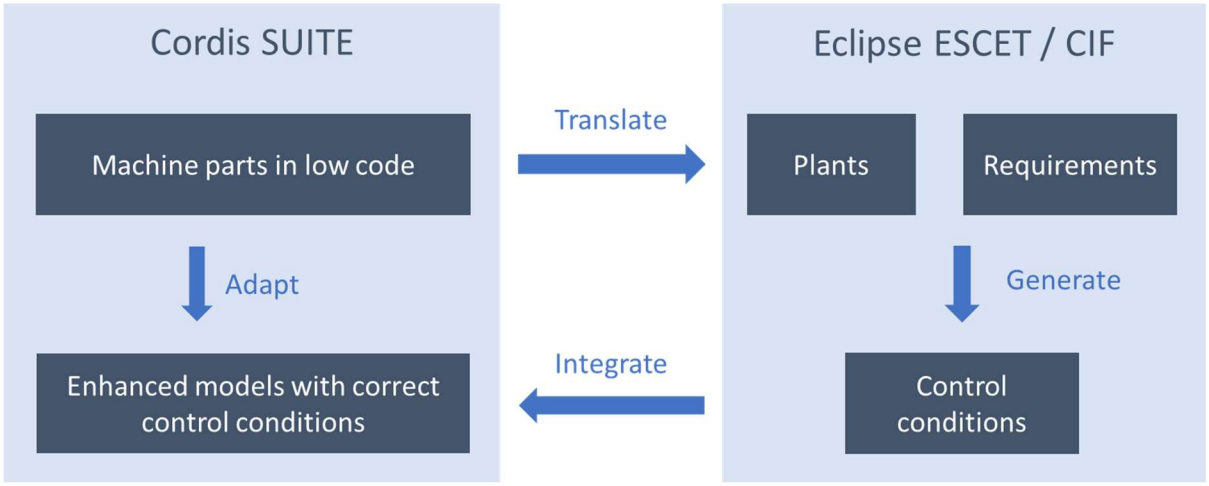

Figure 2: Our workflow for the two pilots. In the second pilot we did not make an enhanced model by integrating the synthesized conditions.

Figure 2 illustrates our workflow for performing the first pilot. Firstly we worked out the pick-and-place system in Cordis SUITE, into a number of machine parts (for the locations and the pick-and-place mechanism), thereby not (yet) considering the FIFO requirement. Secondly we manually translated the low-code specification of the system to a plant model in CIF, which is a modelling language that comes with Eclipse ESCET™ (Fokkink, et al., 2023), which in turn is a toolkit for supervisory controller synthesis.[5] Thirdly we formalized and wrote down the FIFO requirement in CIF, which allowed automatically generating correct control conditions that make the system model satisfy the requirement. The generated conditions for example ensure that not too many products can enter the system (at most three), and that the 'redo' action cannot be taken in a way to make products overtake each other. Finally, we manually translated the generated conditions back to conditions on the Cordis SUITE models, making them correct-by-construction with respect to the FIFO requirement.

To be convinced that the enhanced Cordis SUITE models indeed satisfy the requirements, we developed a simulation environment that randomly interacts with the models (i.e., it randomly lets products enter the system, and randomly performs actions whenever enabled). We showed that, without the generated conditions, the Cordis SUITE models would quickly and consistently get into deadlock situations, or situations that violate the FIFO requirement, often already after a few seconds. However, after having added the generated conditions, no violations were flagged when running the simulator, increasing confidence that they indeed made the system correct with respect to the FIFO requirement.

Although we manually translated Cordis SUITE models to CIF models, and the generated conditions back to conditions in Cordis SUITE, the translation could in principle be automated. The manual translations were mostly mechanic, and we believe they could in principle be fully mechanized.

To conclude, we have shown with this first pilot that low-code development can benefit from SBE, especially to automatically generate parts of the models that are easy to specify as a requirement, but hard to implement, e.g., due to many corner cases and subtle situations that would then have to be considered.

[5] See https://eclipse.org/escet. 'Eclipse', 'Eclipse ESCET' and 'ESCET' are trademarks of Eclipse Foundation, Inc.

6. Second pilot: industrial application of synthesis for low-code

After having demonstrated the benefits of combining synthesis with low-code development, our next aim was demonstrating that such a combination can indeed add value in a real-world industrial context. For this purpose, TNO-ESI together with Additive Industries performed a pilot to apply SBE to guarantee a number of machine damage control requirements in their Cordis SUITE models.

Additive Industries develops metal 3D printers. The software for those printers is specified in Cordis SUITE, as models, from which PLC code is generated. In our pilot, we looked specifically at machine damage control requirements that the Cordis SUITE models must adhere to, to prevent machine damage control. The 3D printers developed by Additive Industries contain various moving components, e.g., to perform powder handling and extraction, and those should never collide since otherwise the printer would damage itself.

A total of six machine damage control requirements were provided by Additive Industries in natural language, expressing that certain machine parts should never collide. Additive Industries was interested in learning whether their Cordis SUITE models indeed adhere to these requirements, and if not, what repairs would then be necessary. The goal of the pilot was establishing whether SBE can help to check and/or establish these requirements, in this low-code setting.

The workflow for this pilot was similar as depicted in Figure 2. We first manually translated the relevant parts of the Cordis SUITE models of Additive Industries to CIF models. Then we attempted to formalize the requirements in CIF, to see what control conditions would be synthesized by CIF synthesis tooling. If no additional conditions would be generated for a requirement, then that would mean that the manually translated CIF model would satisfy the requirement. Otherwise, the synthesized extra condition(s) might be used to repair the CIF model (and therewith possibly also Cordis SUITE models, but we did not translate any generated condition back to Cordis SUITE).

Note that, since we have made a manual translation of the models, any requirement violation may be due to errors introduced by this translation. Moreover, we only translated part of the Cordis SUITE models (i.e., we used a limited scope), and our translation has not been thoroughly tested.

We focus on two machine damage control requirements in particular (since most of the other requirements were similar to the two that we picked). For both these requirements, we have found behavior that could potentially violate the machine damage control requirements (again, modulo any errors introduced in the manual translation). For both requirements, extra control conditions were generated that further restrict possible interaction between the environment and certain machine parts, and further restrict movement control of certain machine parts.

The engineers from Additive Industries that were involved in the pilot could not directly confirm whether the violating behaviors that we found are real behaviors that can indeed occur in practice. On the other hand, they did confirm that being able to automatically find and deal with subtle situations and corner cases is helpful, and the ability to automatically synthesize conditions that can deal with this is valuable design assistance. The engineers indicated that they would be interested in seeing SBE for low-code be applied at other parts of their software system.

7. Conclusion

To conclude, we have demonstrated in two pilots that low-code can benefit from SBE, and that it can

add value in practice. Even though both pilots involved manual translations of models and/or generated conditions, these translations were mostly mechanical and can in principle be automated.

8. References

Fokkink, W., Goorden, M., Hendriks, D., Beek, D. v., Hofkamp, A., Reijnen, F., . . . Vogel, J. (2023). Eclipse ESCET™: The Eclipse Supervisory Control Engineering Toolkit. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)* (pp. 44-52). Paris, France: Springer.

Stramaglia, A., & Keiren, J. J. (2022). Formal Verification of an Industrial UML-like Model Using mCRL2. In J. Groote, & M. Huisman (Ed.), *Formal Methods for Industrial Critical Systems. LNCS, volume 13487*, pp. 86-102. Cham: Springer International Publishing.