Continuous Systems Integration with Virtual Spaces

Architecting for Continuous Software Engineering

Pablo Oliveira Antonino, Thomas Kuhn, Benno Kallweit;
Fraunhofer IESE

Continuous software engineering aims at orchestrating engineering knowledge from various disciplines in order to deal with the rapid changes within the ecosystems of which software-based systems are a part. The literature claims that one means for ensuring these prompt responses is to incorporate virtual prototypes of the system into the development process as early as possible, such that requirements and architecture decisions are verified early and continuously by means of simulations. Despite the maturity of practices for designing and assessing architectures by means of simulations, there is still a lack of platforms that would enable both the design and simulation of architecture drivers and solutions in a continuous engineering context. In this paper we therefore present the Fraunhofer Virtual Space for Continuous Engineering, which is a web-based platform that enables (i) model- and text-based architecture specification (drivers and design), (ii) continuous simulation of architecture solutions against architecture drivers, and (iii) continuous analysis of the integration and consistency of engineering artifacts.

Integration challenges in the embedded systems industry

Industrial reports indicate that difficulties with the integration of new systems into existing systems are key obstacles to business growth [1][2]. It is claimed that the root causes for these integration challenges include rapid changes within the ecosystems of which software-based systems are a part, heterogeneity, amount, and architecture design incompatibility between the existing and the new systems to be integrated. Continuous engineering practices such as continuous integration and continuous monitoring, along with development process trends like DevOps, have been adopted to mitigate these integration challenges [2]. In this regard, Tesla, for example, is efficiently responding to customer requirements requested in social networks [3].

In parallel to this, virtual engineering techniques like digital twins are being used increasingly in the development of embedded systems [4]. The inherent complexity and size of modern systems calls for the configuration, deployment, verification and validation, and even some calibration to be done in virtual representations of the system specified by means of tailored architecture models before the actual deployment of the software to the real hardware.
Despite the maturity of practices for designing and assessing architectures by means of simulations, there is still a lack of platforms that would enable both the design and simulation of architecture drivers and solutions in a continuous engineering context.

To address this challenge, we present the Fraunhofer Virtual Space for Continuous Engineering, (i) model- and text-based architecture specification (drivers and design), (ii) continuous simulation of architecture solutions against architecture drivers, and (iii) continuous analysis of the integration and consistency of engineering artifacts.

Continuous Engineering

Continuous engineering emerged because of the need for a more holistic approach to deal with the rapid changes within an ecosystem into which software-based systems are integrated [1][2][3].

There is a common misunderstanding that continuous software engineering is a synonym for continuous integration or continuous delivery. However, as discussed by Fitzgerald and Stol [2], these are only two of the practices that make up the idea of continuous software engineering, which additionally incorporates aspects intrinsically related to business strategy, development, and operations, such as continuous planning, continuous deployment, continuous evolution, and continuous experimentation. We understand that the proper execution and support of these practices will lead to continuous trust (cf. Figure 1).

Antonino et al. [3] claim that one practice that is a key aspect for properly enabling continuous engineering of embedded systems is the use of Virtual Prototypes. This concept will be discussed in the following session of this paper.

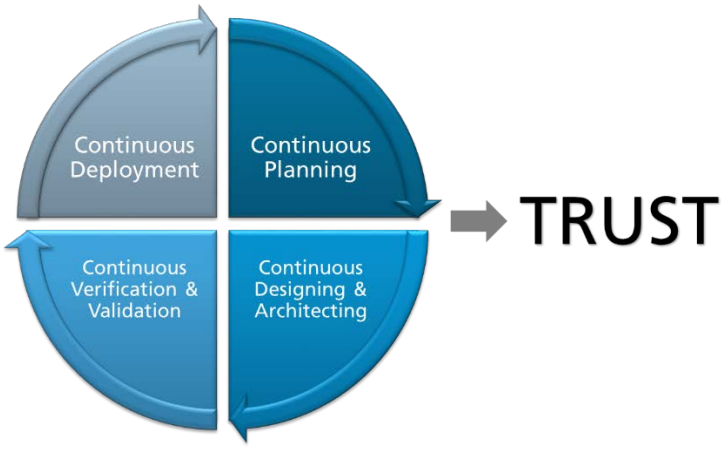


Figure 1. Continuous Engineering Practices.

Virtual Prototypes

Virtual Prototypes (VPs) are the key for enabling continuous integration for embedded systems because they allow performing the concurrent development of usually serial steps (AKA Frontload) and testing the system already in advance in order to detect conceptual problems that cannot be foreseen in the planning phase (AKA Frontloading of testing) [5][6]. VPs can range from the simulation of high-level models to the detailed simulation of processors and other components in Electronic Control Units (ECUs).

Industry reports from Bosch, Hitachi, and General Motors indicate the following advantages and improvements achieved by using virtual prototypes [5]: (i) visibility and controllability across the entire system; (ii) quick correction of specification errors prior to implementation; (iii) early architecture design space exploration; (iv) development and collaboration possible at worldwide locations; (v) physical hardware dependency removed from the supply chain; (vi) development cycles improved by 30% to 50%; (vii) reuse of (parts of) prototypes for future work; (viii) software development starting 9 to 12 months prior to hardware availability; (ix) identification and correction of software bugs within hours rather than days; (x)

integration of new software on first silicon chip within a day; and (xi) more competitive products delivered up to 6 months faster.

The Fraunhofer Virtual Space for Continuous Engineering

The Fraunhofer Virtual Space for Continuous Engineering is a platform that enables continuous specification and continuous evaluation by means of simulations of architecture decisions using virtual prototypes. The platform enables (i) model- and text-based architecture specification (drivers and design), (ii) continuous simulation of architecture solutions against architecture drivers, and (iii) continuous analysis of the integration and consistency of engineering artifacts. Figure 2 depicts an overview of the platform, which is composed of two main components: (i) the Virtual Space Web Front-End and (ii) Fraunhofer FERAL. For the sake of space, this paper will focus on the description of the Virtual Space Web Front-End. Details on the FERAL simulation engine can be found in [7][8].

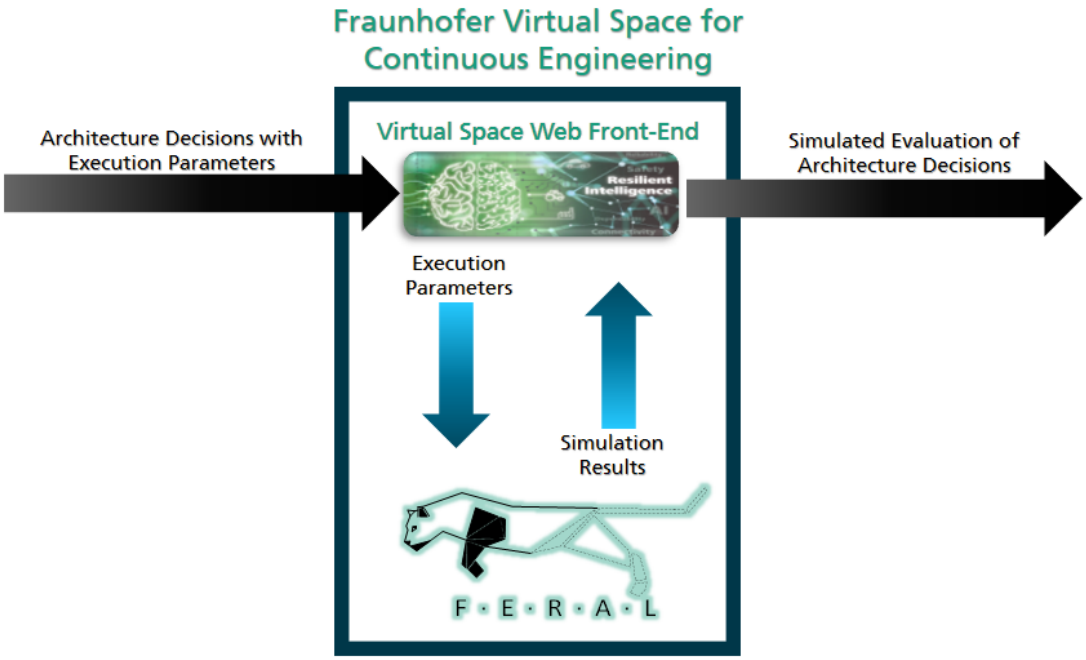


Figure 2. Overview of the Fraunhofer Virtual Space for Continuous Engineering.

The Virtual Space Web Front-End corresponds to the human interaction layer of the platform. It enables the online specification of architecture-significant requirements (AKA architecture drivers) and architecture decisions. This specification is important for continuously evaluating, by means of simulation, the adequacy of the architecture solutions for addressing the architecture drivers. In this regard, every change, addition, or removal at the requirements level or at the architecture level can be continuously verified with the high confidence of simulations.

The specification of architecture drivers can be done using natural or controlled-natural language. An example of architecture drivers specified with natural language is depicted in Table 1, and examples of architecture driver specifications with controlled-natural language using pre-defined templates, such as the Parameterized Safety Requirements Templates, are described in [9].

Architecture Driver Description	
Environment	Vehicle is in update operation mode. Critical application software has been replaced by a compromised one.
Stimulus	Compromised software triggers the initialization of the update.
Response	System security check identifies the malicious software and aborts its initialization.
Response Measure	Counter for malicious software update is incremented in the system log.

Table 1. Architecture Driver Specification Example.

The specification of architecture design decisions can also be done textually or using models. An example of a textual specification is depicted in Table 2, and a diagrammatic representation of the architecture decision in Figure 2. Such models as well as structural architecture descriptions at the functional, logical, and technical (hardware and software) levels can be created in the Virtual Space Web Front-End using tailored embedded systems modeling elements from the Embedded Modeling Profile [10], whose elements are depicted in Figure 3.
These modeling facilities are also available for the modeling tools IBM Rational Rhapsody, Sparx Enterprise Architect, and MagicDraw. It is also possible to create the architecture models in any of these tools and import them later into the Fraunhofer Virtual Space for Continuous Engineering. In cases where the architecture models are created with different modeling languages, it is necessary stereotyping the modeling elements according to the Embedded Modeling Profile metamodel.

Architecture Decision	
Architecture Decisions	- Safety and security code sections must contain integrity checks (CRC, Hash, etc.) and/or be digitally signed, and the system then checks the validity of the CRC or signature. - Cryptographically secure hash for integrity verification, aimed at preventing malicious security threats.
Solution Steps	1. Intruder → VehicleCAN: uploadSoftware(Software Update Binary) 2. VehicleCAN → ECU: CANupdate (Software Update Binary) 3. ECU → ECU: ValidateMessageSignature (MessageSenderID) 4. ECU → SecurityAuditTrail: (SecurityError: MessageSignatureValidationError) 5. SecurityAuditTrail → ECU: (SecurityErrorLogged)

Table 2. Textual specification of architecture decision.

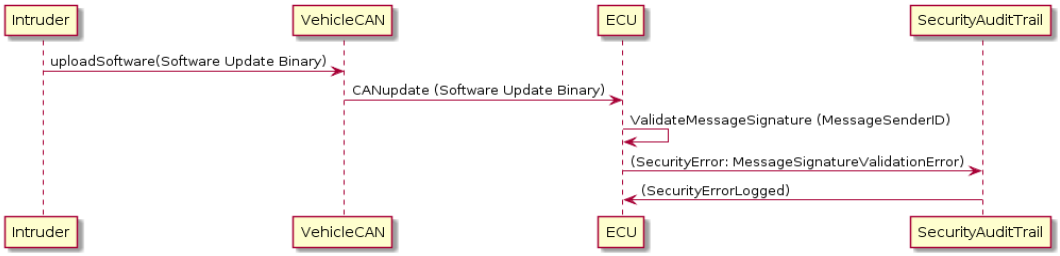


Figure 3. Solution steps of the architecture decision described in Table 2.

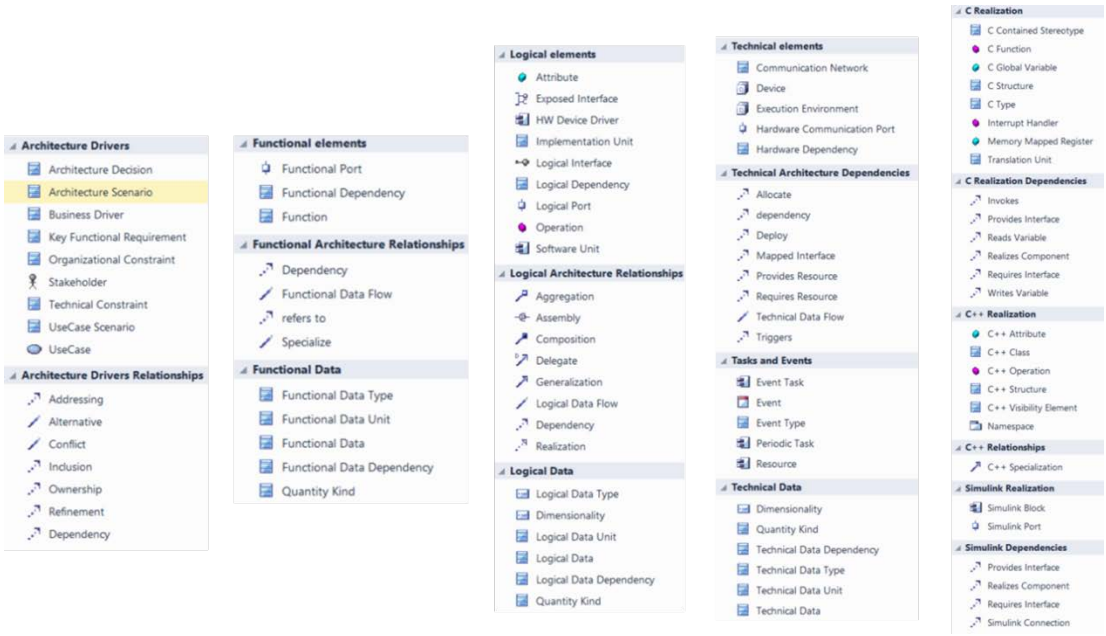


Figure 4. Tailoring Architecture Modeling Elements.

Independent of creating the models directly with the Virtual Space Web Front-End or of importing existing models into the platform, it is possible to specify various parameter combinations to verify the adequacy of the architecture solutions. For example, consider the example illustrated in Figure 4, which depicts the screen for specifying simulation parameters of a Functional Mock-up Unit (AKA FMU). The *General Parameters* portion of the screen is used to specify the general simulation parameters to be executed with Fraunhofer FERAL. The step duration specifies the duration of each simulation step – the other duration specifies the general duration of the overall simulation. In the *Custom FMU Parameters* portion of the screen, custom FMU parameters are automatically read from the description of the .fmu file with their corresponding type. They can be manipulated in this portion of the screen and are passed to the FMU as constants. The *Output Logging* and the *Input Logging* tabs of the *Custom FMU Parameters* portion of the screen are depicted in Figures 5a and 5b. The *Output Logging* portion offers the possibility to select which output port of the FMU shall be logged. Only the ports checked at this level will be shown in the *Simulation Result Section*. The same logic holds for the input parameters. By

pressing the "Start Simulation" button, the specified parameters are sent to the server side of the application. The server side then sets up the necessary FERAL objects, sets the parameter and the corresponding loggers, and starts the simulation. This transformation and the simulation execution are detailed in [7]. Finally, the simulation results of the output variables are displayed in terms of each parameter with the value at each time step.

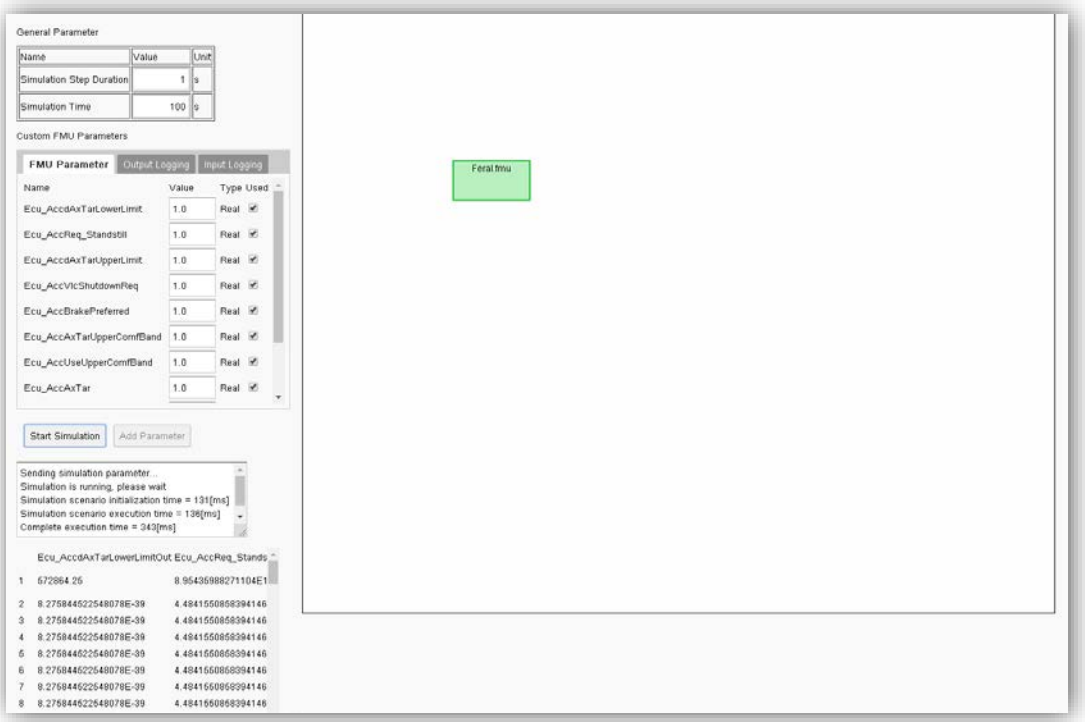
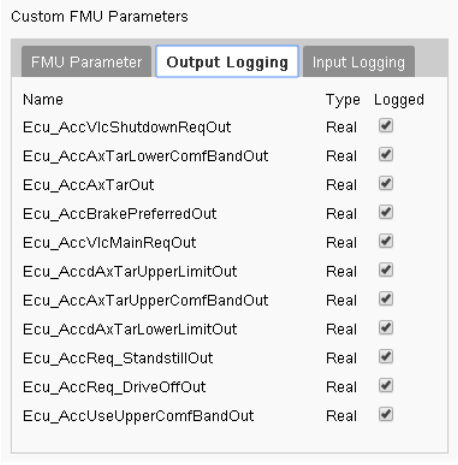


Figure 4. Tailoring Architecture Modeling Elements.



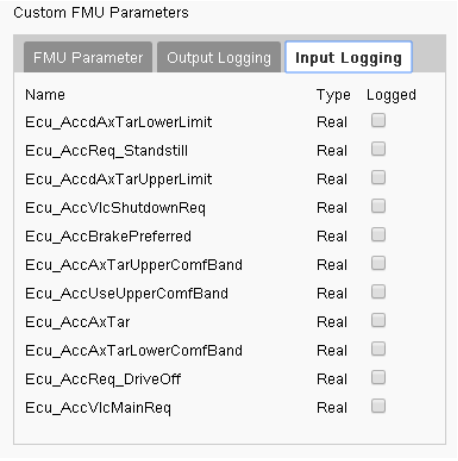


Figure 5a. Output Logging Parameters Section. Figure 5b. Input Logging Parameters Section.

Conclusion

In this paper, we presented the Fraunhofer Virtual Space for Continuous Engineering, which enables the online modeling of architecture specifications and the validation of different system properties at the architecture level by means of simulation with Fraunhofer FERAL. In a nutshell, the platform enables (i) model- and text-based architecture specification (drivers and design), (ii) continuous simulation of architecture solutions against architecture drivers, and (iii) continuous analysis of the integration and consistency of engineering artifacts.

References

[1] C. Shamieh: Continuous Engineering for Dummies. John Wiley & Sons, Inc. (2014).

[2] B. Fitzgerald, K.J. Stol, : Continuous software engineering: A roadmap and agenda. Journal of Systems and Software 123 (2017) 176 – 189.

[3] P.O. Antonino, M. Jung, A. Morgenstern, F. Faßnacht, T. Bauer, A. Bachorek, T. Kuhn, E.Y. Nakagawa: Enabling Continuous Software Engineering for Embedded Systems Architectures with Virtual Prototypes. ECSA 2018: 115-130.

[4] M. Jung, S. Piao, T. Purusothaman, X. Pan, T. Kuhn, C. Grimm, K. Berns, N. Wehn: Virtual Development on Mixed Abstraction Levels: an Agricultural Vehicle Case Study. Synopsys Usergroup Conference (SNUG) (June 2015).

[5] T. De Schutter: Better Software. Faster!: Best Practices in Virtual Prototyping. Synopsys Press, USA (2014).

[6] R.V. O'Connor, P. Elger, P.M. Clarke: Continuous software engineering a microservices architecture perspective. Journal of Software: Evolution and Process 29(11) 2017.

[7] P.O. Antonino, J. Jahic, B. Kallweit, A. Morgenstern and T. Kuhn, "Bridging the Gap between Architecture Specifications and Simulation Models," 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), Seattle, WA, USA, 2018, pp. 77-80.

[8] T. Kuhn, T. Forster, T. Braun, and R. Gotzhein. "FERAL — Framework for simulator coupling on requirements and architecture level," 2013 Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2013), Portland, OR, 2013.

[9] P.O. Antonino, M. Trapp, P. Barbosa, and L. Sousa. 2015. The parameterized safety requirements templates. In Proceedings of the 8th International Symposium on Software and Systems Traceability (SST '15). IEEE Press, Piscataway, NJ, USA, 29-35.

[10] Andreas Morgenstern, Pablo Antonino, Thomas Kuhn, Patrick Pschorn, and Benno Kallweit. 2017. Modeling embedded systems using a tailored view framework and architecture modeling constraints. In Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings (ECSA '17).

Authors

Dr. Pablo Oliveira Antonino is the Head of the Embedded Software Engineering department of the Fraunhofer IESE in Kaiserslautern, Germany. His work focuses on the continuous integration, construction, evaluation, and simulation of architectures of dependable embedded. Dr. Antonino received his PhD from the University of Kaiserslautern – Germany.

Email: pablo.antonino@iese.fraunhofer.de

Dr. Thomas Kuhn received his PhD from Kaiserslautern University in 2009, where he was working as a researcher since 2004. Since 2008, he is working for Fraunhofer IESE. Currently, he is heading the Embedded Systems Division, where applied research and consultancy services are delivered in the areas of smart ecosystems integration, architecture design, evaluation, and simulation, variability management and safety engineering.

Email: thomas.kuhn@iese.fraunhofer.de

Benno Kallweit is research assistant at the Embedded Software Engineering department of t he Fraunhofer IESE in Kaiserslautern, Germany. He studies Computer Science since 2012 with the focus on Software Engineering, and support the development of tools and methods for integration, design, evaluation and simulation of architectures of embedded systems.