# Project Overview for Step-Up!CPS – Process, Methods and Technologies for Updating Safety-critical Cyber-physical Systems

Thomas Strathmann
*Transportation Division*
*OFFIS e.V.*
Oldenburg, Germany
thomas.strathmann@offis.de

Georg Hake
*Department of Computing Science*
*University of Oldenburg*
Oldenburg, Germany
georg.hake@uni-oldenburg.de

Houssem Guissouma
*ITIV Institute*
*Karlsruhe Institute of Technology (KIT)*
Karlsruhe, Germany
houssem.guissouma@kit.edu

Carl Philipp Hohl
*ESS-ESM*
*FZI Forschungszentrum Informatik*
Karlsruhe, Germany
hohl@fzi.de

Yosab Bebawy
*Transportation Division*
*OFFIS e.V.*
Oldenburg, Germany
yosab.bebawy@offis.de

Sebastian Vander Maelen
*Transportation Division*
*OFFIS e.V.*
Oldenburg, Germany
svm@offis.de

Andrew Koerner
*Institute of Transportation Systems*
*German Aerospace Centre (DLR)*
Braunschweig, Germany
andrew.koerner@dlr.de

***Abstract*—We describe the challenges addressed by the three year German national collaborative research project Step-Up!CPS that is currently in its third year. The goal of the project is to develop software methods and technologies for modular updates of safety-critical cyber-physical systems. To make this possible, contracts are utilized, which formally describe the behaviour of an update and make it verifiable at different times of the update life cycle. We have defined a development process that allows for a continuous improvement of such systems by monitoring their operation, identifying the need for updates, and development and deploying these updates in a safe and secure manner. We highlight the points along the update process that are necessary for a secure update and show how we counteract them in a contractually secured update process.**

***Index Terms*—Cyber-physical Systems, Safety and Security, Contract-based Design, Online Monitoring, Variant Management**

## I. Introduction

Cyber-physical systems (CPS) interact independently with their environment to perform planning and control tasks in safety-critical applications such as automated and autonomous driving, networked production facilities or autonomous shipping. The conditions under which these systems perform their tasks can change throughout their life cycle (e.g. through new transport infrastructures for highly automated vehicles, new communication and sensor technologies or extended deployment scenarios). During system development these changes are only partially foreseeable. Therefore, continuous updating, adaptation and expansion of these systems is highly relevant. Future systems must be designed in such a way that improvements based on operational experience can be introduced without endangering their operational safety. Hence, a process for secure, modular updates is needed. This process is backed by foundational research and a generic platform for updatable safety-critical CPS.

This work has been funded by the German Federal Ministry of Education and Research (BMBF) in the project Step-Up!CPS (Förderkennzeichen: 01IS18080D). The responsibility for the content remains with the authors.

This paper presents the collaborative research project Step-Up!CPS (Software methods and technologies for modular updates of Cyber-Physical Systems). The project started in October 2018 and is funded for three years by the German Federal Ministry of Education and Research (BMBF). Under the direction of the OFFIS Computer Science Institute, a consortium consisting of the University of Oldenburg, the DLR Institute of Transport System Technology in Braunschweig, the Karlsruhe Institute of Technology (KIT), the FZI Research Center for Computer Science and SafeTRANS is developing concepts, processes and tools to enable modular updates of safety-critical CPS. The goal is to develop prototypical solutions (TRL 3–4) for the problems outlined in the following section. The research is driven by four use cases from the domains automotive, maritime, and industry 4.0. These use cases are complemented by a generic demonstrator which is described in section III. To ensure the relevance of the research to industrial needs, an Industrial Advisory Board comprising representatives from companies in the various application domains participated in the elicitation of requirements and continues to guide the progress in regular meetings.

### A. Challenges

The project tackles the following research questions:

- How must a safety-critical CPS be developed to allow for updates?
- How can updates be developed and how can their safety be demonstrated?

- How can updates for entire product lines with a high number of variants be developed as efficiently as possible and how can compatibility be ensured?
- How can a system be updated after commissioning without endangering its safe operation?
- How can the update process itself be secured against malicious actors?

### B. Outline of the Paper

The paper is structured as follows: In section II the salient results of the project so far are summarized. Section III gives an example of applying the Step-Up!CPS process. And section IV outlines the work that will be done until the end of the project in October 2021.

## II. Current State of the Project

### A. Proposed Update Process for Cyber-Physical Systems

The methods and technologies developed by Step-Up!CPS originate from the development processes in the automotive, maritime and industrial domains. This process is augmented by additional activities that are necessary for the iterative development and testing of safe and secure modular updates. In order to achieve the high level of assurance needed for safety-critical systems, it relies on formal modeling using contract-based design. This enables early system verification and compatibility checks of introduced modular changes. Both the process and the architecture have a number of variation points, allowing for tailoring to different application domains and scenarios, so that it can be used to develop updates for cars, ships or industrial plants. Key components of our approach are an update decision process, a software and contract development approach, a secure deployment method as well as safety and security mechanisms in the updated devices. This is complemented by a verification approach that allows to efficiently ensure that software, that is being deployed fulfills the standards for its safe application. During verification, the software changes are checked statically with regards to compatibility as well as dynamically within a simulation environment. Due to the existing variability of the systems that can be updated, a variant handling in testing and deployment is integral part of our process. It ensures that software is tested for all systems present and that an update can only be deployed to a device that has a suitable configuration. Furthermore, the contracts enable the monitoring of the correct system functionality at runtime. Finally, a collaboration model between manufacturer, supplier and client operator allows for joint cooperation. An overview of the development process is shown in Figure 1.

*a) Design Process:* The design of the updated component is triggered by either a change in system requirements (e.g. regulatory changes), errors in system behavior or the request for new functionality. It then follows the accepted steps of the V-model in order to update system requirements and architecture. Contracts for verification are modeled, as well as dynamic models for simulation of the updated system.

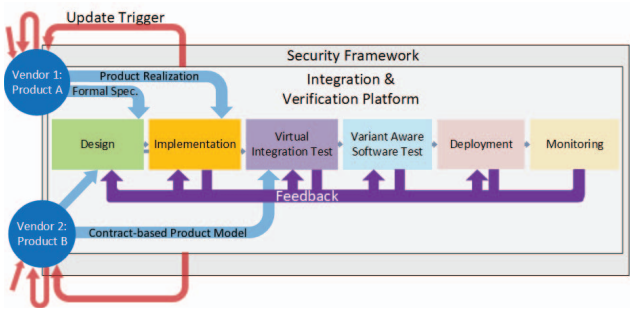


Fig. 1. Overview of the proposed design process steps for updates

*b) Implementation:* The process encourages the use of continuous integration systems to be able to deliver updates on a regular basis. The code and contracts for an updated component are implemented alongside each other and are bundled after build into a package containing the executable and accompanying contracts.

*c) Verification:* During verification, the software changes are checked statically with regards to compatibility as well as dynamically within a simulation environment. The verification of the complete system is done by digital twin testing. This allows to replay data that has been recorded by monitoring in the field in order to ensure a systems safe operation. Within the simulation, the contracts are constantly being checked for violations to detect potential errors in the implementation.

*d) Deployment:* The update package is signed and transmitted to the target system via an encrypted connection. Server and device certificates enable the identification, legitimacy of the parties involved and ensure that only the correct update package is installed. In addition, a Device Configuration Identifier is used to determine that the update may only be updated on the designated variant of the client.

*e) Monitoring:* On the client, monitoring capabilities allow tracking contract violations during operation and generate feedback data by the systems in use. The gathered information can be fed back into development to facilitate the regular initiation of perfective, corrective or adaptive updates.

### B. Contract-Based Design

Contracts are used to formalize requirements and to check whether an update is feasible or not. Contract theory [1] provides the means to (i) derive contracts of the composite functions from its constituents (horizontal process) as well as to (ii) check whether the system functionality is refined by the lower levels of abstraction (vertical process). Each component and each module is equipped with contracts. A module is the smallest updatable unit. A contract comprises an assumption (A) and a guarantee (G). The contract holds when the guarantee is satisfied whenever the assumption holds. Contracts cover the aspects functional, timing, and safety by specifying the behavior of the component under the given assumptions. If a module is updated, the contracts and the results of a Virtual Integration Test (VIT) [2] can be used to

check if the updated module still fulfills the specifications. A mode management concept was developed for fail-operational systems and patterns in terms of contracts were developed to express different operation modes of a system as well as allowed mode transitions in contracts, so that these modes and mode transitions are visible to the VIT. Since the dependency on modes could not be expressed by contracts so far, extended contracts were introduced, which have additional optional assumptions (B) and corresponding guarantees (H). Currently, work is in progress to formally relate the contracts at different design levels by developing an approach that is based on services provided by the execution platform.

### C. Variant Management

One of the most important challenges for developing and validating software updates for CPS is the existing high number of system variants arising from the customization possibilities offered by the Original Equipment Manufacturer (OEM) of the system. In fact, each product evolves both in time ("versions") and space (usually called "variants"). At the same time, it is composed of different components, which in their turn can evolve independently from each other. In Step-Up!CPS, we follow a holistic approach for both variability in time and space by developing corresponding methods in the investigated application domains. These methods enable the verification of each affected CPS variant with a reasonable testing effort to ensure the compatibility of the deployed update packages.

In a first step, we extended the introduced contract-based design approach (cf. section II-B) to cover whole product-lines and not only unique system variants. For this purpose, we chose feature trees as a variability modeling method. Those are used in combination with 150% models of the system architecture, which comprise all existing variability points and their mappings to the related components. This allows us to select any arbitrary variant, to annotate it with contracts and to virtually verify its integration using VIT checks. Then, we introduced in a second step a new incremental verification methodology based on so-called Deltas ($\Delta$), i.e. differences between variants, to reduce the testing effort. The delta concept is based on three different delta categories for changes of interfaces, contracts or implementation (see Figure 2).

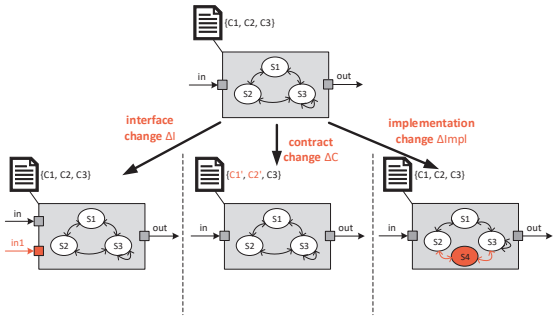


Fig. 2. Introduced delta categories as unit of change between variants

Additionally, we described a concept for the integration of the variant management into the overall process. For the transition from the design to the deployment phase, we specified the needed metadata in form of dependency constraints. These enable an efficient configuration matching between the CPS configurations in the field and the released updates.

Due to the considerable differences between the investigated application domains, we identified a higher need for variant management in the automotive field than in maritime and industry 4.0 domains, where systems are usually unique copies maintained by different stakeholders.

### D. Security Architecture and Validation

The updates are packaged together with contracts and other metadata necessary for deciding deployability. The data is bound together by signatures to be able to detect modification or exchange of package contents. To ensure that no packages from untrusted sources are installed and that no packages can be downloaded to systems that are not eligible, mutual authentication is employed. The mechanism is implemented with HTTPS or SSH depending on the use case. Figure 3 shows an overview of the use of certificates along the development and deployment process.

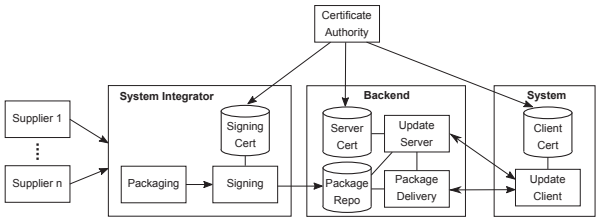


Fig. 3. Use of certificates for securing the update process from development to deployment

For the validation of security architectures we employ a model-based analysis [3]. The functional architecture is annotated with security mechanisms such as integrity protection of components, filtering and routing of messages or encryption. An automatic analysis identifies all the ways an attacker may compromise the security mechanisms. Once this step is completed, the attack scenarios can be combined with a safety view of the system to automatically analyze the impact of attacks on the safe operation of the system.

## III. Use Case

For demonstration purposes, a multi-rotor system has been selected. The multi-rotor is equipped with a controller that determines the thrust of the motors so that the state of the copter corresponds to the user input. In the following we sketch the application of the Step-Up!CPS methodology to this use case.

During the design phase the functional architecture is derived. The multi-rotor controller receives two inputs corresponding to the actual state and the required motions, i.e. the position sensor and the remote controller signals, and produces signals that determine the motors' thrusts. During

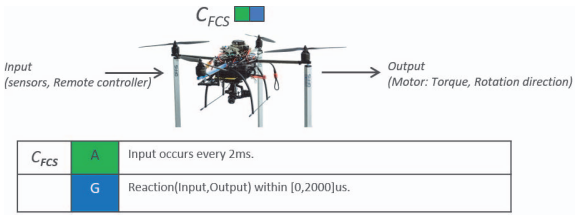


Fig. 4. Multi-Rotor System Requirements

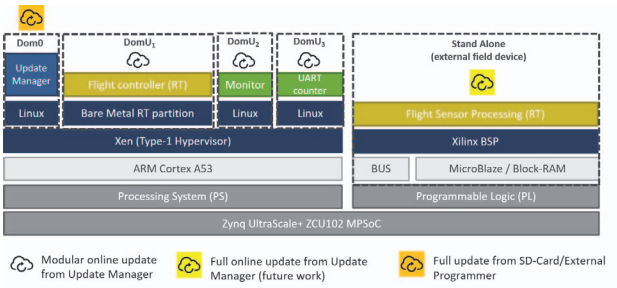


Fig. 5. Multi-Rotor Technical Realization

system (re-)design due to an update request, the system-level requirements must be investigated. These requirements are identified based on the system's as well as environmental constraints, e.g. functional, timing, or safety. The requirements can be formalized as contracts (see Figure 4).

For the variant management part, we implemented the concept in a first prototype by modeling variants of the multi-rotor system. Based on that, we showed how it is possible to identify affected variants for a specific update, to generate representative variant models, and to check them using VIT [4]. Using this approach, it is possible to limit the verification and test effort to a minimum by concentrating only on affected variants and checking specifically the deltas between them.

For design-time verification, the VIT can be (re-)performed in order to check whether refinement relations still hold. A variety of tools (e.g. MULTIC-Tooling [5], OCRA [6]) and methodologies may be applied in order to check different specification aspects. MULTIC supports an automated execution of simulation-based VIT for timing.

For run-time verification, online monitoring components are generated from new requirements. The monitors observe the system's execution in terms of a sequence of events and detect violations of its formal specification (in terms of contracts). Monitors are designed in the same way as functional components, and can be considered as part of the design, including the deployment on the target platform.

The next step is the logical decomposition, where the mapping of the individual functions to modules takes place. Mapping the module to the platform services (e.g. execution, communication, or safety mechanisms) makes it ready for deployment.

During the architecture design the modules are linked to concrete hardware elements such as central processing unit cores, and communication infrastructure as buses and memories (see Figure 5). This includes scheduling of service invocations subject to timing and resource constraints.

The multi-rotor use case has been implemented on a Zynq UltraScale+ development board. The simulation uses a flight simulator which emulates the position sensor and the remote controller inputs and receives the motors values from the Zynq board. The flight simulator also visualized the flight of the multi-rotor system. During testing, a fault was injected into the multi-rotor controller and the integrated monitor was able to detect it. Based on the diagnostic data an update for the controller has been identified. This update request triggered a re-iteration of the design process for the controller.

## IV. Further Work

In the final year work focuses on implementing prototypes of the update mechanism, runtime monitoring, and mode management, as well as some tool support. In most instances this work has already produced intermediate results. The generic demonstrator was key to achieving this given the heterogeneous requirements of the application domains. Each concept, method, and tool will be demonstrated in at least one use case by the end of the project by tailoring the generic concepts to the application domains. The security measures will be implemented in the automotive and the maritime use cases using different technologies. The application of the delta concept for incremental verification of variants is currently being investigated for an exemplary update scenario within one automotive use case. Further implementation options for the maritime and industry 4.0 use case are currently being investigated. For each use case there is a safety concept. As part of these concepts the fail operational mechanisms based on the mode management in conjunction with the runtime monitoring are currently being implemented in the generic demonstrator and one automotive use case.

## References

[1] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen, "Contracts for system design," *Foundations and Trends in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[2] I. Stierand, P. Reinkemeier, and P. Bhaduri, "Virtual integration of real-time systems based on resource segregation abstraction," in *Formal Modeling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science, A. Legay and M. Bozga, Eds. Springer International Publishing, 2014, vol. 8711, pp. 206–221.

[3] T. Strathmann and S. Fröschle, "Towards a model-based safety and security analysis," in *Proc. of the 12th Dagstuhl-Workshop: Model-Based Development of Embedded Systems (MBEES), 15.03. – 17.03.2017, Schloss Dagstuhl, Germany*, M. Huhn, H. Hungar, M. Riebisch, and S. Voss, Eds. fortiss GmbH, 2017.

[4] H. Guissouma, C. P. Hohl, H. Stoll, and E. Sax, "Variability-aware process extension for updating cyber physical systems over the air," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, 2020, pp. 1–8.

[5] W. Damm, G. Ehmen, K. Grüttner, P. Ittershagen, B. Koopmann, F. Poppen, and I. Stierand, "Multi-layer time coherency in the development of ADAS/AD systems: Design approach and tooling," in *Proceedings of the Workshop on Design Automation for CPS and IoT (DESTION'19)*. ACM New York, NY, USA, 04 2019, pp. 20–30.

[6] A. Cimatti, M. Dorigatti, and S. Tonetta, "OCRA: A tool for checking the refinement of temporal contracts." Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, 11 2013.