


Developing an engineering tool for Cyber-Physical Production Systems

U. Kannengiesser , J. Frysak, C. Stary, F. Krenn, H. Müller

Cyber-Physical Production Systems (CPPS) form the basis of the next industrial revolution. However, many manufacturing companies are reluctant to adopt this disruptive technology due to a lack of know-how and a high project risk. In this paper, we present the development of a design tool that addresses common challenges of CPPS engineering, including the complexity of CPPS and the collaborative, multidisciplinary nature of the engineering process. The tool is based on the combination of existing and emerging reference models, standards and methods from software engineering, production automation and Industry 4.0, embedded in a new model of collaborative engineering for CPPS. The paper describes the foundational concepts of the tool, highlights its innovations, and reports some of the insights gained during its development and its usage in an industrial scenario.

Keywords: Cyber-Physical Production Systems (CPPS); Industry 4.0 (I4.0); ISO/IEC/IEEE 42010:2011; Collaborative Software Engineering

Entwicklung eines Engineering-Werkzeugs für Cyber-Physische Produktionssysteme.

Cyber-Physische Produktionssysteme (CPPS) sind die Grundlage für die Vierte Industrielle Revolution. Jedoch geschieht die Einführung dieser disruptiven Technologie in vielen Produktionsunternehmen aufgrund von Know-how-Mangel und hohem Projektrisiko nur zögerlich. In diesem Beitrag präsentieren wir die Entwicklung eines Design-Werkzeugs, um die Herausforderungen im CPPS-Engineering – u. a. die Komplexität von CPPS und den kollaborativen, multidisziplinären Engineering-Prozess – zu bewältigen. Das Werkzeug basiert auf der Kombination von Referenzmodellen, Standards und Methoden aus dem Software Engineering, der Produktionsautomatisierung und Industrie 4.0, eingebettet in ein neues Modell für das Engineering von CPPS. Der Beitrag beschreibt die grundlegenden Konzepte des Werkzeugs, seine neuartigen Charakteristiken sowie die Erfahrungen, die aus seiner Entwicklung und Nutzung in einem industriellen Szenario gewonnen wurden.

Schlüsselwörter: Cyber-Physische Produktionssysteme (CPPS); Industrie 4.0 (I4.0); ISO/IEC/IEEE 42010:2011; kollaboratives Software Engineering

Received May 10, 2021, accepted July 7, 2021, published online July 22, 2021
© The Author(s) 2021



1. Introduction

Cyber-Physical Production Systems (CPPS) are widely seen as the technological basis for the fourth industrial revolution (“Industry 4.0”) – the digitalization and decentralization of production operations, and their integration from shop floor to office floor and across company networks. Industry 4.0 aims to provide greater flexibility, enabling the individualization of products and services to different customers at nearly the cost of mass production. These new technological possibilities can enable entirely new business models, such as lot-size one production and data-driven, predictive maintenance.

CPPS are the application of Cyber-Physical Systems (CPS) in the manufacturing context [14], where CPS denote physical and technical systems whose processes are monitored and controlled by computing units and which can communicate with each other and with other systems (e.g., Internet) through integrated communication units [32]. CPPS are systems that consist of “autonomous and cooperative elements and subsystems that are connected based on the context within and across all levels of production, from processes through machines up to production and logistics networks” [32]. By directly connecting components across production levels, CPPS partially dissolve the hierarchical system levels defined in the traditional automation pyramid (IEC 62264 [20]).

Despite the high business potential of CPPS, many industrial companies remain reluctant when it comes to turning their traditional

production systems into CPPS. The risk associated with introducing this disruptive technology is often perceived as too high, mainly because of the complexity of CPPS and a lack of know-how leading to ineffective outcomes of CPPS engineering and expensive overruns of project schedules.

1.1 Objective

In this paper the development of an engineering support system is described that aims to facilitate the design of CPPS software and thus reduce the risk involved in transforming traditional automated production systems into CPPS. It was carried out as part of a recent R&D project funded by the Austrian Research Promotion Agency (FFG). One of the project partners was a small-sized company specialized in furniture production, which had already automated a few of its shopfloor operations but did not have any experience in CPPS. The company envisioned using a CPPS for the predictive maintenance

Kannengiesser, Udo, Institute of Business Informatics – Communications Engineering, Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria (E-mail: udo.kannengiesser@jku.at); **Frysak, Josef**, Institute for Information Management and Control, Vienna University of Economics and Business, Vienna, Austria; **Stary, Christian**, Institute of Business Informatics – Communications Engineering, Johannes Kepler University Linz, Linz, Austria; **Krenn, Florian**, Compunity GmbH, Linz, Austria; **Müller, Harald**, Compunity GmbH, Linz, Austria

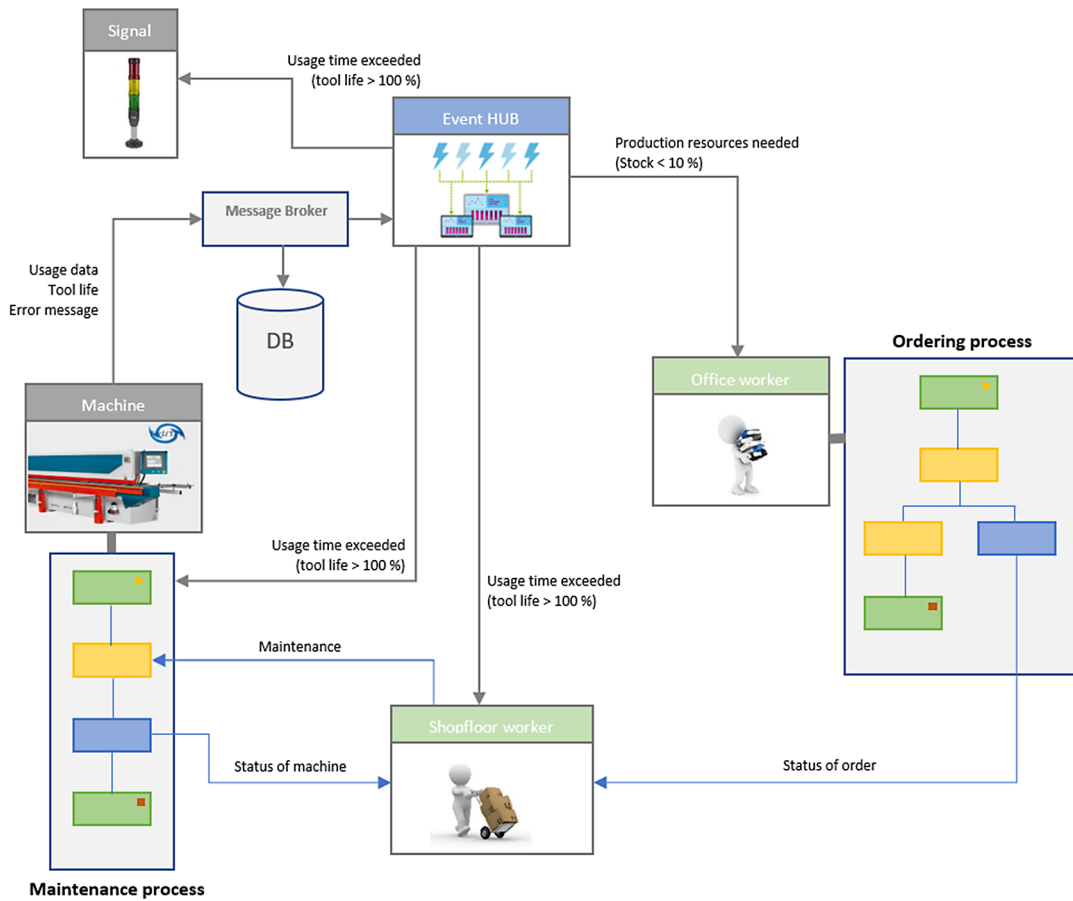


Fig. 1. Predictive maintenance scenario to be realized using a CPPS

nance of a production machine based on sensor data, as shown in Fig. 1. Data on machine utilization, machine life and occurring errors are to be recorded by sensors. Based on this data, the system is to recognize events according to predefined rules, which are then sent to a cloud-based event hub. The event hub analyses the events and triggers physical signals, maintenance processes in production, and ordering processes on the office floor. The aim of the system is to provide maintenance services and related materials, before damage or unplanned downtime occurs.

This paper describes the development of an engineering tool that supports the modelling and implementation not only of the particular system described in Fig. 1 but also any other CPPS. The tool is unique in that it addresses typical challenges in the digital transformation of production processes, which has the potential to lead to more large-scale adoption of cyber-physical technologies in industry.

1.2 Approach

Due to the highly unpredictable and diverse context (e.g., different stakeholders, different industries and domains) of CPPS engineering, we followed an evidence-based approach (cf. [27]) taking into account existing concepts, standards and empirical findings. Evidence is understood as the combination of research, engineering expertise, experiences, and stakeholder preferences. The approach consisted of four steps:

1. **Problem analysis:** Initially, market reports and research reports were reviewed with regard to the challenges of CPPS engineer-

ing. These challenges were then mapped to the specific needs and requirements of a particular application. The general scope of the engineering tool was then determined.

2. **Concept development:** Literature on CPPS, (model-based) software engineering, distributed systems, and business informatics was analysed and relevant concepts extracted. The concepts were then merged into the new approach to software engineering in CPPS. Care was always taken to avoid tailor-made models/concepts by making extensive use of existing standards and reference models in order to constantly maintain relevance to the industry.
3. **Implementation:** Based on the previously developed concepts a prototype was developed. The implementation followed an agile, iterative and incremental development process to react on unexpected problems of the concepts. Particular emphasis was placed on avoiding legacy solutions and monolithic software architectures.
4. **Evaluation:** The prototype developed was evaluated with respect to its requirements. The evaluation should check whether all CPPS modelling needs according to the existing stakeholder concerns were met. This means whether the tool (and the methodology implemented) was able to support the needs of all system modellers and adequately support their task in terms of aspects, functionality and usability.

This evidence-based approach was handled in line with traditional project management methods.

Table 1. Summary of challenges in CPPS engineering

Challenge	Aspects
1. Complexity of CPPS	a) Many components and interactions b) Heterogeneous technologies
2. Collaborative, multidisciplinary nature of CPPS engineering	a) Fragmented domain knowledge b) Informal and unstructured knowledge c) Difficult requirements elicitation d) Limited IT skills e) Distribution of project team across time and space

The paper is structured as follows: Sect. 2 provides a summary of the main challenges related to CPPS engineering. Section 3 reviews existing work on CPPS engineering in the light of these challenges. Section 4 presents the principal solution concepts in terms of CPPS modelling constructs, an engineering process and key implementation technologies. Section 5 summarizes the insights and lessons learned gained from the development and usage of the tool. Section 6 concludes the paper with a summary of innovations and future research.

2. Challenges in CPPS engineering

We conducted a literature review to identify challenges in CPPS engineering that are relevant for this project. Detailed accounts of such challenges along the CPPS lifecycle already exist (e.g., [39]). For the purposes of the R&D project reported in this paper, the scope is limited to those challenges that are most directly relevant for small-sized companies with little expertise in CPPS engineering. The results of our literature analysis are summarized in Table 1 and explained in more detail in the remainder of this section.

One of the challenges in engineering CPPS is the *high complexity* of these systems [14], making their engineering and reliable operation difficult. This is because CPPS are characterized by high degrees of decentralization, with potentially large numbers of components that can act autonomously and interact with one another in ways that are not always predictable. CPPS can thus be regarded as an instance of ultra-large-scale (ULS) systems [7]. In addition, CPPS span multiple layers of the automation pyramid with a heterogeneous set of different technologies and constraints (e.g., real-time behaviour, cf. [29]). Their seamless integration remains a challenge throughout the plant life cycle.

Another challenge is the *collaborative nature* of CPPS engineering, requiring multiple stakeholders from different domains [3]. One problem here is a practical one: How can the stakeholders effectively collaborate when some of them may not even be located in the same place (e.g., across different companies) and need to adhere to different operational work schedules? Another problem is that different stakeholders are used to working with different models that reflect their own perspectives of a system. Often these stakeholders are domain experts with little IT knowledge, thus limiting their ability to use complex engineering or modelling tools. It is rare to find “Industry 4.0 experts” that unify all the required knowledge and skills of CPPS engineering within the same person, despite various calls for dedicated higher education courses [1]. Even in cases when such an expert is available, usually in the role of an external consultant, there is still the problem that the specific operations, requirements and business context within a production company remain opaque unless this knowledge is elicited from that company’s personnel.

The two challenges – the complexity of CPPS, and the collaborative, multidisciplinary nature of the engineering process – provide principal dimensions for assessing existing CPPS engineering approaches and for driving the development of a new approach. Some of the specific aspects characterizing the challenges are also summarized in Table 1. They address the variety of components and interactions to be considered for CPPS engineering, and related hindrances for effective development. Heterogeneous technologies lead to interoperability issues. Fragmented domain knowledge due to the nature of diverse CPPS expertise leads to collection and alignment effort of engineering knowledge. Informal and unstructured knowledge requires codification steps for accomplishing CPPS engineering tasks. Limited IT skills represent obstacles for the effective participation of all stakeholders and require considerable learning effort. Distribution of project teams across time and space is due to different geographic locations of production sites and engineering stakeholders, and leads to coordination effort. Requirements elicitation for CPPS is cumbersome, since there exist different perceptions of requirements terminology and wording, representing barriers for effective engineering processes.

3. Background and related work

In this section we review relevant approaches proposed specifically for CPPS engineering based on cyber-physical systems characteristics and corresponding development experiences. We focus on model-driven engineering, since it has been identified as most crucial for meeting identified development challenges [39].

3.1 CPPS development

CPPS development as any other CPS requires strategies to handle a highly heterogeneous process, since various stakeholders working on a development artefact need to interact albeit different tools and engineering processes due to tight integration of heterogeneous technologies [6]. Decisions made by one stakeholder not only have an impact on the system design but also on the behaviour of other stakeholders. Specifications concerning different subsystems capture hardware, software, and physical aspects. Hence, meeting the challenges of complexity and collaboration in CPS engineering has been recognized as socio-technical endeavour including social guidelines and legal regulations (cf. [26, 35]).

Engineering support needs to target integrated computational and physical capabilities, in particular the dynamic collection of data in the physical world and its processing (in the cyber world) ultimately changing the physical world through actuators. Most existing methodologies have been adapted from software and embedded systems development, among them the V-Model, Model-based development, and Agile Development (cf. [33, 40]). For instance, the V-Model (cf. [10, 30]) structures engineering according to phases representing a logical sequence of steps, relying on stepwise verification. However, the V-Model does not tackle the aforementioned complexity of development processes and their collaborative nature. Hence, its application requires substantial adaptation effort, e.g., utilizing the SPES 2020 method [37]. It enables structuring the design phase (left branch of V-Model), addressing views on requirements, functions, logics, and technology. It has been extended with references to the physical components and human behaviour by [11]. Such structural views help meeting the complexity challenge due to their separation of concerns.

3.2 Model-based engineering

Model-based software engineering (MBSE) as also introduced for CPPS development [2] tackles the formal specification and/or visualization of design models focusing on technology, such as learning-enabled components [13]. Using models, stakeholders are capable

of articulating their needs to be used by designers and engineers. In order to express engineering knowledge (domain) modelling languages are used. The Systems Modeling Language (SysML) is based on the Unified Modeling Language (UML) and has been used along intertwined user- and technology-centred development steps [2].

For instance, Merlo et al. [31] identify three phases, starting with exploration, elaborating on top-down designs. The second phase concerns synthesis with respect to selection and unit tests, and the third phase, termed convergent bottom-up design, development. The latter need to be detailed through decomposition to trigger the selection of best fit of concepts in phase 2. It is achieved through prototyping and scenario-based testing. The validation of the final concept in phase 3 is followed by detailing design and prototyping before final experiments to validate the whole product.

Knowing the semantics of a (domain-specific) modelling language allows for executable models, enabling code generation and integration (cf. [16]). They enable validation, simulation, and finally, seamless roundtrip engineering. Thereby, the quality of representation and the consistency between models are tested. Although constructive refactoring and stepwise evolution of models enriches the agility, corresponding development approaches, such as [33] based on Scrum, still requires engineers from different disciplines to collaborate when aligning the interplay of various digital and physical components.

Based on SysML, [8] introduced a language to link disparate yet overlapping engineering models as part of a framework to resolve inconsistencies of engineering models. It captures the specification of consistency rules by using dedicated types of links. In order to capture model relations, three layers of abstractions have been introduced, organized bottom up: Links, Linking Language, and Link Type Definition Language. Links are represented in a Link Model. It contains the definition of application-specific links between distinct models. It lays ground to the Linking Language capturing the application- or domain-specific language. The Link Type Definition Language is a generic language, allowing to define application- or domain-specific link meta-models.

A prototypical case study showed that the framework can handle heterogeneous engineering models, thus, aligning not only models specified in various types of languages (general purpose, tool-specific, task-specific, domain-specific) but rather preparing for automated production engineering based on consistent representations. One current limitation is the acquisition of rule-specific knowledge by, from and for CPPS engineers (albeit the diagrammatic modelling effort). However, rules seem suitable for representing model relations and further processing when adjusting engineering models.

These findings in model-based engineering show again the need for structuring the alignment of engineering knowledge, and for providing means to express relationship in a semantically accurate way. These relationships need to be set dynamically along CPPS engineering processes, while operating on fundamental engineering structures.

4. Solution concepts

In this section we start with detailing the constructs for CPPS engineering models we have utilized for a viewpoint-oriented approach in conformance with Industry 4.0 concepts. Subsequently, an engineering process supported by the proposed tool is presented, with which the CPPS models can be generated and transformed into executable Industry 4.0 components.

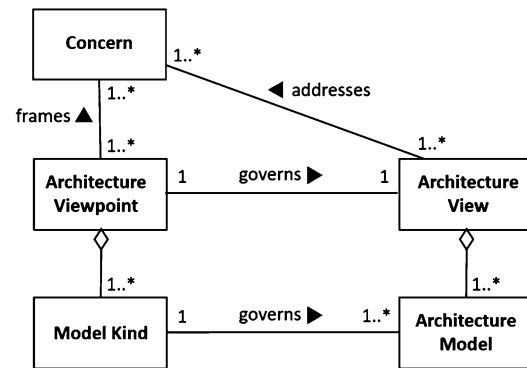


Fig. 2. Key elements of ISO/IEC/IEEE 42010:2011 used for the CPPS engineering tool

4.1 Constructs for CPPS model

Representing complex systems is facilitated by separating and encapsulating the various concerns and views of a system. This is one of the approaches of research in software and system architecture. A common effect of the various existing architectural models [4] is that they enhance understanding and development of a system. The ISO/IEC/IEEE 42010:2011 [21] standard provides a conceptual framework for the architectural modelling constructs used for the CPPS engineering tool [23]. The principal constructs adopted from this standard are shown in Fig. 2.

Of particular importance for the present work are the notions of architecture views and architecture viewpoints. An architecture view (or short: “view”) is defined in the standard as a “work product expressing the architecture of a system from the perspective of specific system concerns”. An architecture viewpoint (or short: “viewpoint”) is defined as a “work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns”. The distinction between a view and a viewpoint can be likened to the distinction between a map and a legend (that provides the conventions for producing and reading the map). A viewpoint is represented using a set of “model kinds” that provide meta-models governing the construction of architecture models composing the corresponding view (e.g., by means of data flow diagrams, Petri nets, state machines, etc.).

Viewpoints for CPPS architectures are consistent with reference models currently emerging in the industry. One of the most widely known reference models has been developed by the German “Plattform Industrie 4.0” that is among the leading CPPS standardization initiatives worldwide. It includes a layered architecture for so-called “Industry 4.0 components” (or short: “I4.0 components”) whose structure and behaviour are currently being defined as an IEC standard (IEC PAS 63088:2017 [18]). I4.0 components are composed of an “asset” (i.e., any physical or virtual object of value for a company, including human operators, production machines, documents, workpieces, etc.) and an “asset administration shell” (AAS) that encapsulates the asset [36], as shown in Fig. 3.

The AAS provides a set of functionalities for standardized communication with other I4.0 components and for representing and managing the digital twin of the asset. The digital twin is a computational representation of the asset, organized in the following layers:

- Business layer: that includes interactions and behaviours of I4.0 components and various contextual factors from the business environment

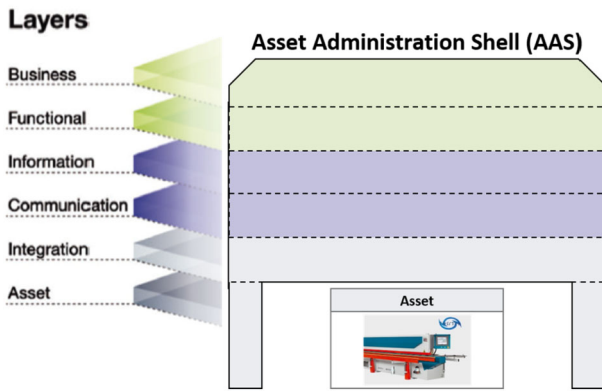


Fig. 3. Architecture of an I4.0 component

- Functional layer: that includes a component's functions and services
- Information layer: that includes various data models describing the component and its interfaces
- Communication layer: that includes standard interfaces for message exchange between I4.0 components
- Integration layer: that includes digital signals from the physical world

For modelling CPPS in the sense of MBSE, not all of these layers are required. The hardware-specific Integration layer is assumed to be provided by the asset vendors. Models in the Communication layer can be generated by export into standard message protocols such as OPC UA. It is only the models in the Business, Functional and Information layers that need to be created by domain experts. Often a viewpoint relevant to a domain expert involves all three layers. For example, a viewpoint relevant to a machine operator is likely to include conventions for constructing or interpreting technical data specifications (Information layer), machine functions (Functional layer), and processes in which the machine plays a role (Business layer).

For each of the three modelling layers, highly generic model kinds were developed aiming to lower the barrier for untrained domain experts to create their individual views of the CPPS. These model kinds can be extended to include more specific meta-data that constrain and add details to the architecture models created. The generic model kinds include constructs and conventions for:

- *Data modelling* (Information layer): The only mandatory attributes for a data element include name and type. Additional attributes, such as regular expressions, ranges of values, and visualization options, can be defined as desired. Modellers can create hierarchical data structures or flat lists of data elements.
- *Service modelling* (Functional layer): Services are modelled in terms of interfaces merely consisting of input and output parameters. This conceptualization matches the notion of function blocks in the IEC 61131-3 [17] standard for PLC programming. Input and output parameters are references to data models in the Information layer. Concrete service implementations can be added and linked to the service interface.
- *Skill modelling* (Business layer): Skills represent the behaviour of the I4.0 component and how this behaviour interacts with skills provided by other I4.0 components. Technically, skills are modelled as state machines, where states represent either services provided internally by the I4.0 component or send/receive actions for external interaction [9].

Architecture models produced according to these viewpoints are called "submodels", in conformance with the I4.0 literature [36].

4.2 CPPS engineering process

Research in engineering design has shown that mismatches exist between prescriptive models of design processes and the actual behaviour of designers [22]. Designers frequently deviate from such models, as they explore the design space in opportunistic ways [12, 28] depending on specific project constraints, individual problem-solving strategies and emerging design issues. The tool developed for CPPS engineering is therefore based on a process model that allows stakeholders to flexibly adapt their design behaviour to the situation at hand.

A model that fulfils this requirement has been proposed by Oppl [34]. Although it was developed for the domain of process modelling, it is sufficiently general to be applied in a broader context including CPPS engineering. This is because it does not prescribe any particular sequence of artefact-related activities and outcomes in the way typical stage-gate ("waterfall") approaches do. Instead, it focuses on the collaborative aspect of the process, proposing phases of concurrent articulation of individual models alternating with phases of collaborative consolidation. This conceptualization is an implicit assumption of many models of design including the V-Model [38] and other models on the spectrum from stage-gate to agile. This assumption was made explicit in [7].

A CPPS engineering process adapted from Oppl's [34] work, together with different modules of the CPPS engineering tool, are presented in Fig. 4. It can be seen as a subset of common systems engineering lifecycles, covering the phases of system design, realization and validation related to software aspects [39]. The process begins with the phase *Setting the Stage* as a typical project management activity, in which the problem is analysed and the engineering task is scoped. Relevant concerns for the specific CPPS engineering task are identified in this phase, and a set of viewpoints consistent these concerns are defined. In addition, relevant stakeholders are determined and corresponding access rights to the engineering tool are provided. An "Administration" module of the CPPS engineering tool supports the control of access rights and the definition of viewpoints in terms of model kinds.

The next phase, *Individual View Modelling*, comprises individual modelling activities by multiple stakeholders, which can be executed concurrently. They can use predefined, domain-specific model kinds (e.g., database model kind) or utilize a generic model kind that only requires typed data elements without any particular structure. The current version of the tool allows every stakeholder to create I4.0 submodels consisting of data models, services, and skills.

Once a number of views (i.e., I4.0 submodels) have been defined, stakeholders can enter the phase of *Collaborative Consolidation* and align their individual models. This can include the creation of mappings between data models across different viewpoints, e.g. mappings between a data model in an SAP system and the corresponding data model in an external SQL database. Different views may also be consolidated on the level of skills (i.e., on the RAMI 4.0 Business layer), by connecting the skills of two I4.0 components via corresponding messages. For example, a message required for invoking the execution of a skill by component A needs to be connected with a corresponding message sent by component B. The modelling editor of the CPPS engineering tool supports both, the individual view modelling and the *Collaborative Consolidation* phase. Users can arbitrarily switch between the two phases. The result of the modelling phase is an

aligned system model containing and interlinking all I4.0 components that the involved stakeholders agree upon.

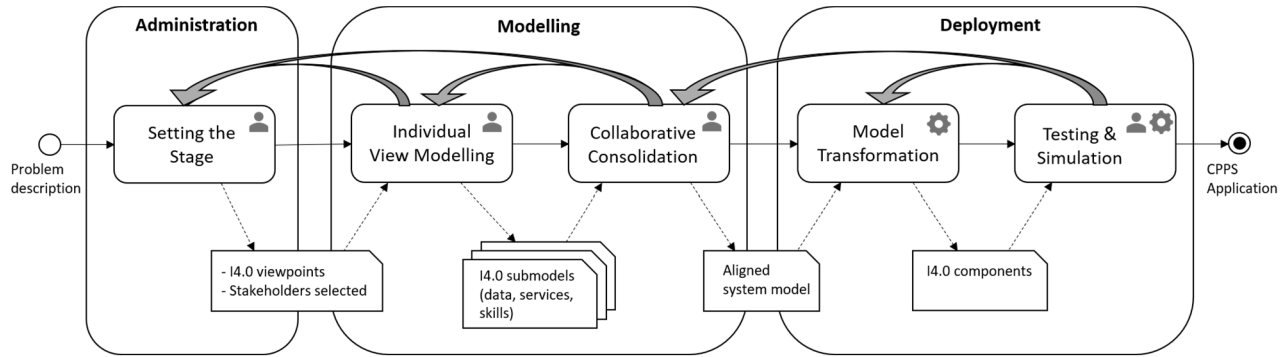


Fig. 4. CPPS engineering process (adapted from Oppl (2016)): Process phases are grouped into modules of the engineering tool

Table 2. Mapping the challenges in CPPS engineering to characteristics of the solution developed

Challenge	Aspects	Solution characteristics
1. Complexity of CPPS	Many components and interactions	Architectural viewpoints to reduce the perceived complexity for individual stakeholders
	Heterogeneous technologies	Use of industry standards and reference models to enable interoperability
2. Collaborative, multidisciplinary nature of CPPS engineering	Fragmented domain knowledge	Different viewpoints can be defined for different types of components and different engineering technologies; Support for creating mappings between data models and skill models across different viewpoints
	Informal and unstructured knowledge	Direct involvement of all domain experts to capture and structure their concerns and work practices; Support for formalizing requirements through definition of suitable viewpoints
	Difficult requirements elicitation	Incrementally accumulating common ground on CPPS requirements by iterating between individual view modelling and collaborative consolidation; Traceability of requirements and models based on the tool's support for all phases of the engineering process
	Limited IT skills	Use of simple, high-level modelling constructs; simple UI that can be tailored to different user roles
	Distribution of project team across time and space	Support for concurrent engineering based on separation of concerns (viewpoints) and independent modelling of the individual views

The following *Model Transformation* phase builds upon the aligned system model. This model is automatically transformed into executable I4.0 components including the defined workflows and message flows. The generated I4.0 components can then be deployed in a given runtime environment (RTE). Different transformation/code generation algorithms for different RTEs (cf. Sect. 4.3) can easily be added. The resulting I4.0 components (i.e., source code) are then used as input to testing and simulation.

The *Testing & Simulation* phase operates on the executable I4.0 components. A static model checking algorithm identifies possible lifelocks and deadlocks in a given CPPS configuration and visualizes where and when they can occur. Skills and CPPS configurations can also be tested using so-called "test scenarios". This feature allows users to model certain test scenarios and expected outcomes similar to unit tests. These tests are then converted to source code and executed. The results are reported back to the modelling tool.

The icons in the top right corner of every phase in Fig. 4 indicate whether it is performed using manual or automated activities. It can be seen that the downstream phases of model transformation and, partially, testing & simulation are automated. This allows system models to be readily deployed, enabling the use of rapid prototyping for validating and verifying the engineered CPPS. The

"upstream" arrows in the figure indicate the possibility of returning to previous phases in the engineering process, resulting in iterations and reformulations of the design space.

Table 2 summarizes how the challenges identified in Sect. 2 have been addressed by the framework presented.

4.3 Solution and prototype implementation

The technological foundation for the tool prototype has been mainly influenced by (1) the architectural concepts proposed by the "Platform Industrie 4.0" and (2) the goal to achieve as much flexibility with the resulting I4.0 components as possible. Analysing the resulting requirements the Actor Model of computation was identified as well suited [5, 15]. The actor model allows designing fine-grained, concurrent systems using message-passing as the sole means to exchange information between the system components. This approach is in large parts congruent with the underlying concepts of the Plattform Industrie 4.0. Therefore, a technical solution very close to the proposed concepts could be realized. In addition to the conceptual fit, production-ready and well-supported frameworks exist facilitating the implementation of a system following the Actor Model. In the prototype the.NET version of the Akka framework

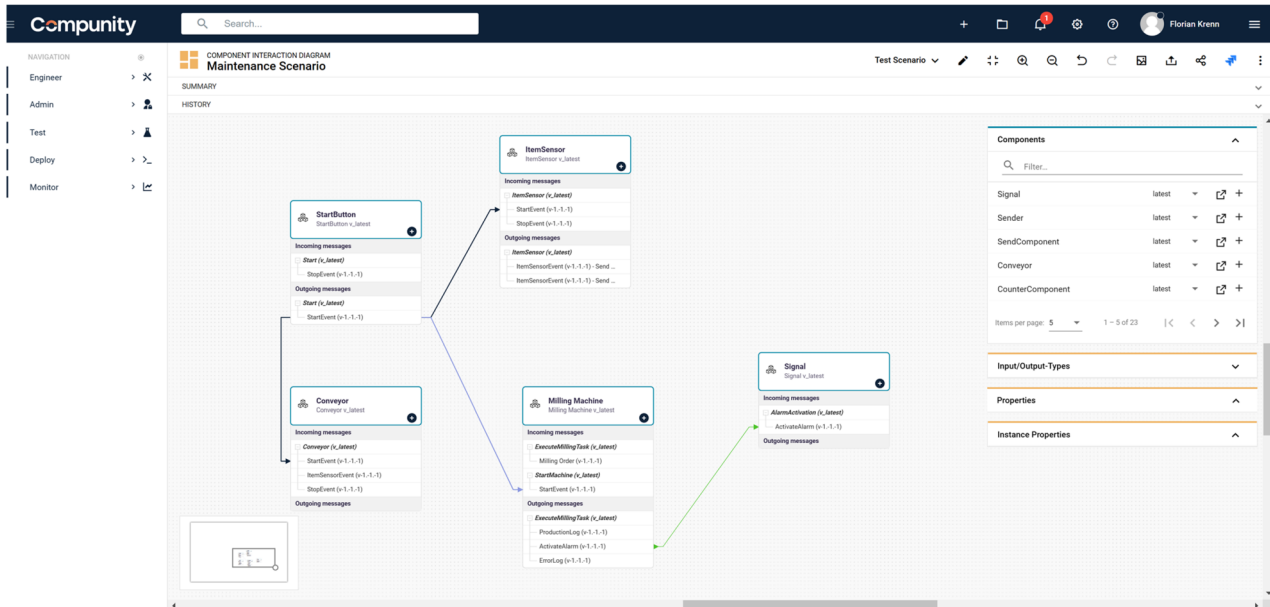


Fig. 5. Component Interaction Diagram (simplified) for the predictive maintenance scenario

(<https://getakka.net/>) was used, which is the most popular actor framework currently available.

In order to support production-specific environments, additional code generation functionality has been added. The data models and service interfaces can be exported as OPC UA structures. This export functionality facilitates a consistent use of data structures throughout a production environment.

In the remainder of this section, the implementation details of a selected feature set are described.

The modelling UI has been implemented as a responsive web application. The UI and modelling concepts have been developed based on some of the concepts provided by Plattform Industrie 4.0 (cf. Asset Administration Shell, RAMI 4.0) and the subject-oriented modelling approach "PASS".¹ Figures 5 and 6 show screenshots of the modelling editor used for producing a CPPS configuration and a skill model, respectively, for the maintenance scenario introduced in Fig. 1.

All changes to model elements (from simple data models to complex I4.0 configurations) are tracked and logged by the implemented versioning system. The versioning features support modellers in two ways. Firstly, every change done to a modelling element can be undone/redone. Errors can conveniently be corrected and modelling decisions can easily be changed. Secondly, using the so-called "time slider" the evolution of the model over time can be visualized. This allows modellers to revise certain decisions or to show new team members how the model has evolved.

The modelling environment offers several synchronous and asynchronous collaboration features to support engineering teams. A major challenge in collaborative modelling is to avoid inconsistencies and conflicts (e.g., two team members change the same part of a model concurrently). In the current implementation synchronous collaborative modelling is supported for the system configuration phase where I4.0 components are interconnected to form the CPPS. Several users can join a modelling session. Each participant is assigned a certain colour. Changes made to the model by a user are

instantly visualized for every user in the corresponding colour. A locking mechanism is implemented to prevent modelling conflicts. Once a user selects a modelling component it is locked to prevent other users from interacting with it. This is visualized by highlighting the locked object with the colour of the user who selected and locked the modelling element.

To reduce the risk of semantic inconsistencies, additional features have been implemented. I4.0 components can be marked for requiring explicit approval when used in a CPPS configuration. For this purpose, a component owner can be assigned to every I4.0 component. If such a component gets added to a CPPS configuration and is linked to other components by a user other than the component owner, the component owner is notified and can approve or decline the use of the component.

The current prototype uses C#/ .NET as target platform of the code generation module. Round-trip engineering features allow for changing data structures, skill sequences, and service/submodel implementations and reimporting them in the graphic modelling tool. This feature is built based on the Microsoft Roslyn platform. Roslyn provides a rich feature set for source code parsing and analysis. The roundtrip features use this library to compare the existing source code tree with the expected result based on the source model and infer the changes made in the code. The identified changes are then converted to modelling actions and applied to the graphic models. This results in a new version of the model ensuring a continuous model history and versioning.

5. Insights gained

Various insights were gained as a result of this project. Some of them were based on the usage of the tool for engineering the predictive maintenance system outlined in Sect. 1 (cf. screenshots in Sect. 4.3). Others were drawn from the numerous experiences made during ideation, concept development and prototype implementation.

5.1 Insights from using the tool

The following insights were gained from the usage of the tool in the predictive maintenance scenario shown in Fig. 1:

¹ <http://i2pm.net/category/interest-groups/standardisation/>.

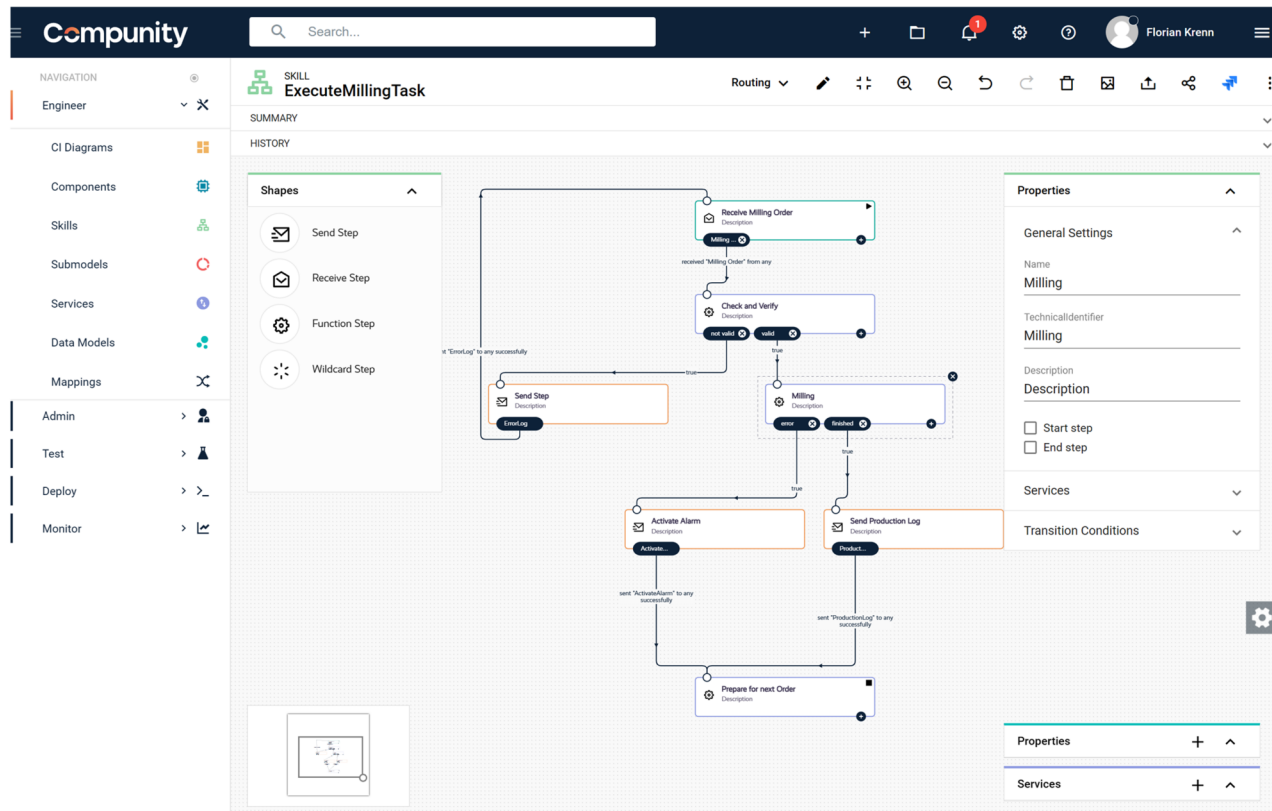


Fig. 6. Skill model (simplified) for a production machine to be maintained

Sufficient coverage of stakeholder concerns: Initial CPPS models for the scenario were developed and aligned by three domain experts: a business analyst, an automation engineer and a C# developer. The models were transformed into executable representations to be deployed on the furniture company's shopfloor. In interviews conducted after the engineering task, all three domain experts reported that the constructs that were provided for data modelling, service modelling and skill modelling have generally covered all their relevant concerns. The viewpoints they had access to allowed them to produce models at sufficient levels of detail for generating deployable CPPS software. This confirms the validity of the design assumptions of the modelling editor and underpinning reference model for I4.0 components that underpin these assumptions.

Genericity vs. domain-specificity: Despite the coverage of stakeholder concerns being sufficient for performing the engineering task, the high genericity of the viewpoints was seen critically by the domain experts. This is because the various constructs and rules reflecting domain-specific conventions and modelling practices were not incorporated in the viewpoints provided for the scenario. For example, the automation engineer and the C# developer were missing specific PLC viz. C# data structures, regular expressions, constructs for unit testing. While it is technically possible to extend viewpoints in a way that they can capture these aspects, the general issue of trading off genericity and domain-specificity has been identified as requiring more research.

Tool complexity: One of the goals in the development of the engineering tool was to reduce tool complexity to a minimum. Yet, the modelling behaviour and some of the interview statements of users indicate that there is still potential for improvement in this respect. This is despite the use of highly generic viewpoints and the resulting

simplicity of the views, e.g. by using service models consisting solely of an input and an output. Some of the issues identified include difficulties related to orienting and navigating within the same model and between different models. This is partly caused by the remaining complexity of the system to be modelled, albeit the separation of concerns supported by the tool. More effective user experience design is likely to be needed to make the visualization of models and tool functionalities more intuitive. This may also address the issue that basic IT knowledge and software skills are still needed in the present version of the tool, which may represent an obstacle for involving untrained domain experts.

5.2 Insights from developing the tool

The following insights were gained during the project phases of concept ideation/development and prototype implementation:

Difficulties dealing with meta-models: Incorporating research in systems and software architecture, including its various abstractions for viewpoint-oriented modelling (ISO/IEC/IEEE 42010:2011 [21]) in the CPPS engineering tool turned out to be quite difficult. Thinking along several levels of abstraction, as required by the distinctions between viewpoints (model kinds) and views (models), was recognized at times to be a non-trivial cognitive task. The problem was exacerbated in project meetings when different team members were thinking and communicating at different levels of abstraction, sometimes leading to long, strenuous and ineffective discussions. Despite the difficulties, however, the understanding of the ISO/IEC/IEEE 42010:2011 constructs and their inclusion in the engineering tool eventually succeeded and was found effective for reducing the perceived complexity of CPPS for the tool user and worth the development effort.

Limitations of existing standards: Standardization processes are generally slow, which also holds for the development of CPPS / I4.0 standards and reference models. Therefore, the basic strategy of aligning concept development with existing norms often reached its limits when models needed to be developed for which no mature standard was available. In many cases, best guesses based on the latest I4.0 whitepapers and on own scientific and practical experiences had to make up for incomplete or ambiguously specified (pre-)standards such as for I4.0 components. One example of such a best guess is the “skill modelling” used for describing the behaviour of an I4.0 component. It is not part of any I4.0 standard but has been identified as an emerging concept in relevant literature, which is already well established in the robotics field. The project team believes that the notion of skills will become part of future I4.0 standards. More generally, this insight supports the view that standardization and innovation can cross-fertilize each other: Standardization may support industry-wide adoption of standard-compliant innovations, while innovative solutions (once proven successful and eventually becoming mainstream) may drive future standardization.

Difficulties choosing suitable technologies: High-tech, research-intensive areas of software development such as CPPS are often characterized by a wealth of whitepapers and informal models but a lack of formal standards and publicly available implementations. As a consequence, there is high uncertainty regarding the choice of technologies to transform high-level architectures and conceptual models into working solutions. Effective risk management is crucial, and rework due to technological decisions turning out to be suboptimal needs to be factored in. In the project described in this paper, OWL was initially chosen as a technology for representing and storing CPPS models (and model kinds), based on its potential support for “intelligent” CPPS behaviours through semantic reasoning processes (as advocated by several research papers in the literature). Yet, prototype tests have shown that the performance of the tool rapidly decreased with the number of models stored. Therefore, the decision was taken to switch from OWL to an on-premise, document-oriented database system that does not store any data schemas, which significantly increased the performance. OWL is still used as an exchange format for CPPS models. One lesson to be learned from this experience, especially for participants in CPPS standardization bodies, may be to increase their focus and development activities on reference implementations that can guide adopters of I4.0 standard models more effectively.

5.3 Lessons learned

The insights and experiences gained from using and developing the tool have led to a few adaptations in the way we carry out our current and future R&D projects. Firstly, we realised that the approach we chose (cf. Sect. 1.2) was not sufficiently agile in the sense that iterative development and rapid prototyping was limited to the implementation phase. There were a number of late changes, such as in software technologies (cf. Sect. 5.2), that could have been prevented if the knowledge gained from implementation had been available earlier. The sources of knowledge explored in the initial phases of problem analysis and concept development (e.g. I4.0 reference models) were indispensable but could not provide information at a level of detail to be directly useful for implementation. Finally, the importance of simple, intuitive user interfaces was underestimated. Given the high complexity of CPPS, any additional complexity caused by the tool is to be kept to a minimum. UI design has therefore become a prime concern in our current R&D projects.

6. Conclusion

Cyber-Physical Production Systems are widely seen as the basis for the next industrial revolution. While research in CPPS architectures and CPPS-based business models abounds, less work exists in establishing effective support systems for practitioners that may reduce the risk involved in transitioning to an Industry 4.0. Amongst the biggest challenges for CPPS engineering are the high complexity of these systems and the need for collaboration of multiple experts from different domains. The tool described in this paper was designed with the goal to address these issues. Some of the foundational concepts for developing the tool include established approaches from software engineering such as MBSE, generative development and viewpoint-based architectures. They are combined with Industry 4.0 reference architectures and a model of collaborative engineering, resulting in a new way of CPPS modelling that is characterized by a high level of flexibility and automation.

The research and development reported in this paper is innovative in several ways:

Viewpoint-based CPPS engineering: In this R&D project the idea of viewpoints as specified in ISO/IEC/IEEE 42010:2011 [21] has been brought into an engineering tool for CPPS. While a few concepts from this standard (including viewpoints, concerns, stakeholders) have been incorporated in the Industrial Internet Reference Architecture (IIRA) [19] – a model for CPS applications in a wide range of industries – it has not been used for CPS/CPPS modelling as comprehensively as described in the present work.

Consequent use of industry standards: To enhance the adoption of the tool by practitioners, its development was closely oriented to the use of existing, industry-proven standards, such as OPC UA and IEC 61131-3 [17] and the emerging norms for I4.0 reference models. Albeit the current gaps in the I4.0 standards landscape discussed in Sect. 5, this approach is likely to increase confidence in the quality of the solution, the possibility of integrating other tools, and the independence from legacy solutions (i.e., avoiding “vendor lock-in”). This differentiates the tool developed in this project from most commercial CPS platforms and academic prototypes.

Flexible engineering process: The CPPS engineering process presented in Sect. 4.2 is highly flexible as it can incorporate different viewpoints and different styles of modelling. Users can freely navigate between individual modelling and collaborative consolidation phases, without being restricted to any rigid sequence of phases with fixed deliverables and decision gates. This leads to increased applicability and wider acceptance by the users.

Implicit practices are made explicit: Most engineering processes in practice are performed by teams rather than one individual. The collaboration between the team members, however, usually occurs implicitly, i.e. not supported or captured by any computational design system. The tool developed in this research makes this collaboration explicit, by integrating it as a formal phase in the engineering process and supporting it by functionalities for mapping the different views. This enhances collaboration, traceability and leads to a more seamless integration of the different views.

The work reported here opens up a number of possibilities for future research and development, including:

- *Process extensions:* A framework may be developed for extending the CPPS engineering process model to match the specific requirements and constraints of the individual CPPS project. It may provide users with more guidance to proceed through the engineering process. This can include refining the process for various engineering methods, including stage-gate and agile techniques.
- *Viewpoint hierarchies:* Viewpoints may be structured hierarchically using inheritance relationships to better enable reuse across

different CPPS projects and thus accelerate the engineering process. Such an approach may also facilitate establishing an open marketplace for viewpoints, where vendors of I4.0 components can develop and offer specialized viewpoints as extensions of more general, standard-conform viewpoints.

Follow-up projects are already under way to expand the functionalities of the tool. They concentrate on integrating concepts from agile software engineering [24] and semi-automated support for configuring I4.0 components [25]. These developments are part of an overall vision for CPPS engineering to become more stakeholder-oriented, incremental and reactive to change.

Acknowledgements

The research reported in this paper is funded by Austria's Research Promotion Agency (FFG) via project no. 862006 (SICHTEN 4.0).

Funding Note Open access funding provided by Johannes Kepler Universität Linz.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Die in diesem Artikel enthaltenen Bilder und sonstiges Drittmaterial unterliegen ebenfalls der genannten Creative Commons Lizenz, sofern sich aus der Abbildungslegende nichts anderes ergibt. Sofern das betreffende Material nicht unter der genannten Creative Commons Lizenz steht und die betreffende Handlung nicht nach gesetzlichen Vorschriften erlaubt ist, ist für die oben aufgeführten Weiterverwendungen des Materials die Einwilligung des jeweiligen Rechteinhabers einzuholen. Weitere Details zur Lizenz entnehmen Sie bitte der Lizenzinformation auf <http://creativecommons.org/licenses/by/4.0/deed.de>.

References

- Baygin, M., Yetis, H., Karakose, M., Akin, E. (2016): An effect analysis of industry 4.0 to higher education. In 2016 15th international conference on information technology based higher education and training (ITHET) (pp. 1–4).
- Berardinelli, L., Mazak, A., Alt, O., Wimmer, M., Kappel, G. (2017): Model-driven systems engineering: principles and application in the CPPS domain. In Multi-disciplinary engineering for cyber-physical production systems (pp. 261–299). Cham: Springer.
- Biffi, S., Lüder, A., Gerhard, D. (Eds.) (2017): Multi-disciplinary engineering for cyber-physical production systems: data models and software solutions for handling complex engineering projects. Cham: Springer.
- Bruneliere, H., Burger, E., Cabot, J., Wimmer, M. (September 2017): A feature-based survey of model view approaches. *Softw. Syst. Model.*
- Carl, H. (1977): Viewing control structures as patterns of passing messages. *Artif. Intell.*, 8(3), 323–364.
- El-Khoury, J., Asplund, F., Biehl, M., Loiret, F., Törngren, M. (2013): A roadmap towards integrated CPS development environments. In 1st open EIT ICT labs workshop on cyber-physical systems engineering.
- Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Northrop, L., Schmidt, D., Sullivan, K., Wallnau, K. (2006): Ultra-large-scale systems: the software challenge of the future. Pittsburgh: Software Engineering Institute.
- Feldmann, S., Kernschmidt, K., Wimmer, M., Vogel-Heuser, B. (2019): Managing inter-model inconsistencies in model-based systems engineering: application in automated production systems engineering. *J. Syst. Softw.*, 153, 105–134.
- Fleischmann, A., Kannengiesser, U., Schmidt, W., Stary, C. (2013): Subject-oriented modeling and execution of multi-agent business processes. In 2013 IEEE/WIC/ACM international joint conferences on web intelligence (WI) and intelligent agent technologies (IAT) (pp. 138–145).
- Forsberg, K., Mooz, H. (1991): The relationship of system engineering to the project cycle. In INCOSE international symposium 1 (Vol. 1, 57–65).
- Grimm, M., Anderl, R., Wang, Y. (2014): Conceptual approach for multi-disciplinary cyber-physical systems design and engineering. In Proceedings of TMCE 2014, Budapest, Hungary.
- Guindon, R. (1990): Designing the design process: exploiting opportunistic thoughts. *Hum.-Comput. Interact.*, 5, 305–344.
- Hartsell, C., Mahadevan, N., Ramakrishna, S., Dubey, A., Bapty, T., Johnson, T., Koutsoukos, X., Sztipanovits, J., Karsai, G. (2019): Model-based design for CPS with learning-enabled components. In Proceedings of the workshop on design automation for CPS and IoT (DESTION'19) (pp. 1–9).
- Hehenberger, P., Vogel-Heuser, B., Bradley, D., Eynard, B., Tomiyama, T., Achiche, S. (2016): Design, modelling, simulation and integration of cyber physical systems: methods and applications. *Comput. Ind.*, 82, 273–289.
- Hewitt, C., Bishop, P., Steiger, R. (1973): A universal modular ACTOR formalism for artificial intelligence. In Proceedings of the 3rd international joint conference on artificial intelligence (IJCAI'73) (pp. 235–245).
- Höldobler, K., Michael, J., Oliver Ringert, J., Rumpe, B., Wortmann, A. (2019): Innovations in model-based software and systems engineering. *J. Object Technol.*, 18, 1–60.
- IEC (2013): IEC 61131-3:2013 programmable controllers - Part 3: programming languages. 3.0nd ed.
- IEC (2017): IEC PAS 63088:2017 smart manufacturing - reference architecture model Industry 4.0. (RAMI4.0). 2nd ed.
- IIC (2017): The industrial Internet of things volume G1: reference architecture.
- International Electrotechnical Commission (2013): IEC 62264 enterprise-control system integration. 2nd ed. Geneva: IEC.
- ISO/IEC/IEEE (2011): ISO/IEC/IEEE systems and software engineering – architecture description. In ISO/IEC/IEEE 42010:2011(E) revision of ISO/IEC 42010:2007 and IEEE std 1471-2000, December 2011 (pp. 1–46).
- Kannengiesser, U., Gero, J. S. (2017): Can Pahl and Beitz' systematic approach be a predictive model of designing? *Des. Sci.*, 3, e2017.
- Kannengiesser, U., Müller, H. (2018): Towards viewpoint-oriented engineering for industry 4.0: a standards-based approach. In 2018 IEEE industrial cyber-physical systems (ICPS) (pp. 51–56).
- Kannengiesser, U., Krenn, F., Stary, C. (2020): A behaviour-driven development approach for cyber-physical production systems. In 2020 IEEE conference on industrial cyberphysical systems (ICPS), 2020 (pp. 179–184).
- Kannengiesser, U., Krenn, F., Stary, C., Höfler, P. (2020): Tindustry: matchmaking for I4.0 components. In M. Freitag, A. Kinra, H. Kotzab, H. J. Kreowski, K. D. Thoben (Eds.), Subject-oriented business process management. The digital workplace – nucleus of transformation, S-BPM ONE 2020. Communications in computer and information science (Vol. 1278). Cham: Springer.
- Kiel, D., Müller, J. M., Arnold, C., Voigt, K.-I. (2017): Sustainable industrial value creation: benefits and challenges of Industry 4.0. *Int. J. Innov. Manag.*, 21, 8.
- Kitson, A., Harvey, G., McCormack, B. (1998): Enabling the implementation of evidence based practice: a conceptual framework. *BMJ Qual. Saf.*, 7(3), 149–158.
- Lawson, B. (2006): How designers think. London: Routledge.
- Lee, E. A. (2008): Cyber physical systems: design challenges. In 2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC) (pp. 363–369).
- Mc Caffery, F., Casey, V., Sivakumar, M. S., Coleman, G., Donnelly, P., Burton, J. (2012): Medical Device Software Traceability. In J. Cleland-Huang, O. Gotel, A. Zisman (Eds.), Software and Systems Traceability (pp. 321–339). London: Springer.
- Merlo, C., Abi Akle, A., Llaría, A., Terrasson, G., Villeneuve, E., Pilnière, V. (2019): Proposal of a user-centred approach for CPS design: pillbox case study. *IFAC-PapersOnLine*, 51(34), 196–201.
- Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihm, W., Ueda, K. (2016): Cyber-physical systems in manufacturing. *CIRP Ann.*, 65(2), 621–641.
- Mulder, F., Verlinden, J., Maruyama, T. (2014): Adapting scrum development method for the development of cyber-physical systems. In Proceedings of TMCE 2014, Budapest, Hungary.
- Oppl, S. (2016): Articulation of work process models for organizational alignment and informed information system design. *Inf. Manag.*, 53(5), 591–608.
- Pagliari, L., Mirandola, R., Trubiani, C. (2019): Engineering cyber-physical systems through performance-based modelling and analysis: a case study experience report. *J. Softw. Evol. Process*, 32(1), e2179.
- Plattform Industrie 4.0. (2016): Structure of the Administration Shell. Plattform Industrie 4.0.
- Pohl, K., Hönninger, H., Achatz, R., Broy, M. (Eds.) (2012): Model-based engineering of embedded systems: the SPES 2020 methodology. Berlin: Springer.
- VDI-Fachbereich Produktentwicklung und Mechatronik (2004): VDI 2206 - Design methodology for mechatronic systems. VDI-Gesellschaft Produkt- und Prozessgestaltung.
- Vogel-Heuser, B., Fay, A., Schaefer, I., Tichy, M. (2015): Evolution of software in automated production systems: challenges and research directions. *J. Syst. Softw.*, 110, 54–84.
- Zheng, C., Le Duigou, J., Hehenberger, P., Bricogne, M., Eynard, B. (2016): Multidisciplinary integration during conceptual design process: a survey on design methods of cyber-physical systems. In DS 84: proceedings of the DESIGN 2016 14th international design conference (design), 1625–1634.

Authors

**Udo Kannengiesser**

is a professor in business informatics at Johannes Kepler University Linz, Austria. He has an educational background in mechanical and production engineering, and obtained a PhD in Design Computing & Cognition from the University of Sydney (Australia). Prior to joining JKU he worked for several software companies, research institutes and universities in Australia, Germany and Austria. He has published around 90 research papers in the areas of design research, process management, multi-agent systems, and digital manufacturing. Among his principal research contributions is an extension of the "Function-Behaviour-Structure (FBS) framework" that is one of the most highly cited models of designing.

**Josef Frysak**

is currently employed at the Department of Digital Economy at the University of Applied Sciences for Management & Communications in Vienna (FH-Wien of WKW). He holds a doctoral degree in Business Informatics from Vienna University of Economics and Business (WU) and his research interests range from software development to Industry 4.0, decision support systems, operations research and decision-making. He holds various courses on UML-Modeling, Business Information Systems and Digital Transformation at Universities and Universities of Applied Sciences in Austria. Currently, he is doing research in the area of Industry 4.0 and the view-based design of Cyber Physical Systems (CPS).

**Christian Stary**

holds a PhD (1988) and a Diploma (1984) in usability engineering from the Vienna University of Technology (TU Wien), where he completed his Habilitation (venia doctendi) in 1993. He is currently a full professor in Business Information Systems at the JKU Department of Business Informatics-Communications Engineering. He also chairs the JKU Knowledge Management Competence Center. At JKU he is Department Head and Study Admission Chair. Christian Stary is the Chair of ICKM (International Council of Knowledge Management). His research interests are work knowledge elicitation and socio-technical modeling, and organizational learning support and development. He has published several books, papers in journals and archival proceedings on these topics.

**Florian Krenn**

received his PhD in Business Informatics from the Johannes Kepler University in Linz. His research activities focused on complexity and understandability of (business) process models. Currently, Florian Krenn is Managing Director of compunity GmbH where he coordinates the company's research and development activities.

**Harald Müller**

received his diploma from the Vienna University of Economics and Business. Besides his profound management skills, Harald Müller is an expert in the field of software development project management and IT infrastructure consulting. After several years as CTO of an insurance company, he founded the compunity GmbH in 2018 where he currently holds the position of Managing Director.