Industry Voices on Software Engineering Challenges in Cyber-Physical Production Systems Engineering

Kevin Feichtinger[1*], Kristof Meixner[3], Felix Rinker[3], István Koren[4], Holger Eichelberger[5], Tonja Heinemann[6], Jörg Holtmann[7], Marco Konersmann[8], Judith Michael[9], Eva-Maria Neumann[10], Jérôme Pfeiffer[6], Rick Rabiser[1,2], Matthias Riebisch[11], Klaus Schmid[5]
[1]LIT Cyber-Physical Systems Lab, Johannes Kepler University Linz, Austria
[2]CDL VaSiCS, LIT Cyber-Physical Systems Lab, Johannes Kepler University Linz, Austria
[3]CDL SQI, Institute of Information Systems Engineering, Technische Universität Wien, Austria
[4]Chair of Process and Data Science, RWTH Aachen University, Germany
[5]Software Systems Engineering, Institute of Computer Science, University of Hildesheim, Germany
[6]Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Germany
[7]Department of Computer Science and Engineering, Chalmers | University of Gothenburg, Sweden
[8]Institute for Software Technology, University of Koblenz-Landau, Germany
[9]Software Engineering, RWTH Aachen University, Germany
[10]Institute of Automation and Information Systems, Technical University of Munich, Germany
[11]Software Development and Software Construction Methods, Universität Hamburg, Germany
[*]E-Mail: kevin.feichtinger@jku.at

***Abstract*—Cyber-Physical Production Systems (CPPSs) are envisioned as next-generation adaptive production systems combining modern production techniques with the latest information technology. A CPPS creates a complex environment between different domains (mechanical, electrical, software engineering), requiring multidisciplinary solutions to tackle growing complexity issues and reduce (maintenance) effort. Software plays an increasingly important role in assuring an effective and efficient operation of CPPSs. However, software engineering methods applied for CPPSs seem to lag behind modern software engineering methods, where tremendous progress has been made in the last years. We initiated the Software Engineering in Cyber-Physical Production Systems Workshop (SECPPS-WS) to analyze and overcome this gap. After two instances with mostly academic participants, we conducted a full-day workshop with nine industry representatives from eight companies that develop and maintain CPPSs. Each industry representative presented their current work and challenges. We collected these challenges and condensed a categorized list of challenges backed by industry statements and literature. This paper presents the resulting list and pointers to (partial) solutions to offer guidance for academia and identify promising research opportunities in this area.**

***Index Terms*—Digital Transformation, CPPS Engineering, Research Challenges**

I. Introduction

In Cyber-Physical Production Systems (CPPSs) engineering, engineers from different disciplines (e.g., mechatronic, electronic, and software engineering) develop modern production systems that manufacture highly customizable goods tailored to customer needs [1]. CPPSs adapt to uncertain conditions of their physical environment using the latest information and communication technology [1], [2]. In practice, CPPS engineering raises several challenges, which require multidisciplinary solutions to tackle growing complexity issues and reduce engineering and maintenance efforts. These challenges also concern the engineering of software for CPPSs and its operation.

Several works discussed CPPS challenges [3], [4] and presented research agendas to address them [5], [6]. For instance, Bureŝ et al. [7] summarized challenges for software engineering in Cyber-Physical Systems (CPSs) and outlined the importance of flexible approaches to address changing requirements, uncertain conditions, structural transformations, and evolving CPSs. Vogel-Heuser et al. [4] investigated the evolution impact and challenges on the life-cycle phases of developing automated production systems and outlined a research roadmap to address these challenges. In particular they focused on software, which gains an increasingly important role in the development of effective and efficient CPPSs. Modern software engineering approaches, such as agile methods, microservice architectures, continuous integration and deployment, and variability modeling, could help. However, based on the authors' experience and recent empirical results by Berger et al. [8], it does not seem that these approaches made their way into industry nor into CPPS engineering.

In 2021, we initiated the Software Engineering in Cyber-Physical Production Systems Workshop (SECPPS-WS)[1] [9] to investigate the topic in more detail and grasp important challenges for software engineering in particular. The workshop aims to discuss approaches and methods to develop software for CPPS and address challenges in adopting state-of-the-art software engineering tools and techniques to the CPPS

978-1-xxxx-xxxx-1/22/$31.00 ©2022 IEEE

[1]https://rickrabiser.github.io/secpps-ws/

domain. After two workshops with participants mostly from academia, we invited industrial stakeholders to a third instance to present their ongoing work and point out current challenges from their industrial practice.

In this paper, we present and discuss the results of this workshop as a categorized list of CPPS engineering challenges underpinned by industry voices. We back these challenges using related literature and outline partial solution candidates. The paper does not provide a systematic overview of the literature and solution candidates. However, we argue that the industry voices combined with the collected literature and solution candidates provide a good foundation for future research directions. Similarly, while the list of challenges might not be complete, it provides a broad overview of what industry is currently struggling with.

The remainder of the paper is structured as follows. Section II describes how we elicited and consolidated the challenges. Section III presents and discusses the categorized list of challenges. Section IV discusses lessons learned and outlines a research agenda. Section V concludes the paper.

II. Research Method

This section presents how we elicited the challenges in the SECPPS-WS and consolidated them for this work utilizing the methodology of a judgement study [10].

After the second SECPPS-WS, we established a reading group to get an overview of recent literature from the field and to collect (partial) solutions to challenges described in the literature. We used the discussions among reading group participants as a basis for questions to industry partners, which we invited to the third SECPPS-WS instance. We invited industrial stakeholders to present their ongoing work and point out current challenges from their industrial practice in CPPS software engineering. Nine practitioners from eight European companies of different size developing and maintaining CPPSs or solutions for CPPSs participated in the workshop. The range of domains reached from automotive manufacturing over medical supply engineering to solution providers and integrators for CPPSs. While the stakeholders might not cover the entire CPPS industry, we carefully selected the practitioners, thus covering a representative set of experts in the field [10].

All academic participants took notes based on the practitioners' presentations and discussions among participants in a shared document. Additionally, we later received feedback on CPPSs challenges from one industry partner who was unable to attend the workshop and integrated this feedback into our notes. The authors of this paper then discussed the notes and elicited concrete challenges for software engineering in CPPS in multiple meetings. Specifically, we used open coding [10] to highlight challenges in notes and created a mindmap from these codes to discuss and refine the challenges among all authors until we achieved a common understanding of the challenges. We also used the results of the reading group to assess how existing work might tackle some of the challenges.

III. Challenges

This section describes the categorized challenges and discusses partial solutions to them. We have explicitly highlighted industry voices in the challenge explanations and set them in context to software engineering research. Fig. 1 shows the categorization of the challenges, also represented by the subsections of this section, and their sub-challenges.

A. Complexity

Complexity in software engineering for CPPS has been reported along multiple dimensions:

System complexity. Two workshop participants reported on the system complexity, which is on the one hand induced by the sheer **amount of different subsystems** and on the other hand by their **heterogeneity** (e.g., sensor systems, real-time control systems, process optimization systems, ...). Correspondingly, Serpano [11] states that singular CPSs encompass traditional IT systems with Operational Technology (OT) systems. These two subsystem classes differ regarding the aspects "purpose, computing components, communication technologies, and interfaces to ownership and management" [11]. Thus, this heterogeneity imposes challenges (i.e., software integration/upgrade, network interoperability, synchronization regarding real-time processes and applications, and security [11]) for the integration of the IT and OT subsystems of one "simple" CPS. Beyond that, a CPPS constitutes a complex *System-of-Systems*, in which the IT subsystems have to control a variety of heterogeneous OT subsystems like Programmable Logic Controllers (PLCs) with software deployed to them. These, in turn, have to interact with each other in a synchronized way. In this context, engineers also have to consider the goods to be produced and the required resources while carefully designing the production process to produce meaningful goods safely.

Time complexity. One company from the special operation vehicle domain reported that CPPSs and their subsystems tend to be **long-living over multiple decades**. This longevity challenges the maintenance and evolution of CPPSs [4] and software engineering [12]. Several workshop participants pointed out **different lengths of development cycles** of the disciplines involved in CPPSs engineering drive these challenges. Mechanics and automation hardware tend to have longer cycles, wheres software tends to have shorter cycles [13]. One participant from the plant-building domain outlined the need for **continuous operation of the plant** (24/7/365). Consequently, experiments in production regarding changing requirements and different development life cycles are impossible. Yet, CPPSs need to evolve over a system's lifespan to maintain their quality, still remain **backwards compatibility** on the hard- and software side. Modularity and well-defined interfaces would help but raise the complexity as interface specifications change, especially in software engineering.

Integration complexity. The complexity of the manufacturing domain requires different stakeholders, everyone having their own domain-specific knowledge and view on the system and services. Thus, different artifacts realizing each view exist. Specifying systems and their services is necessary to

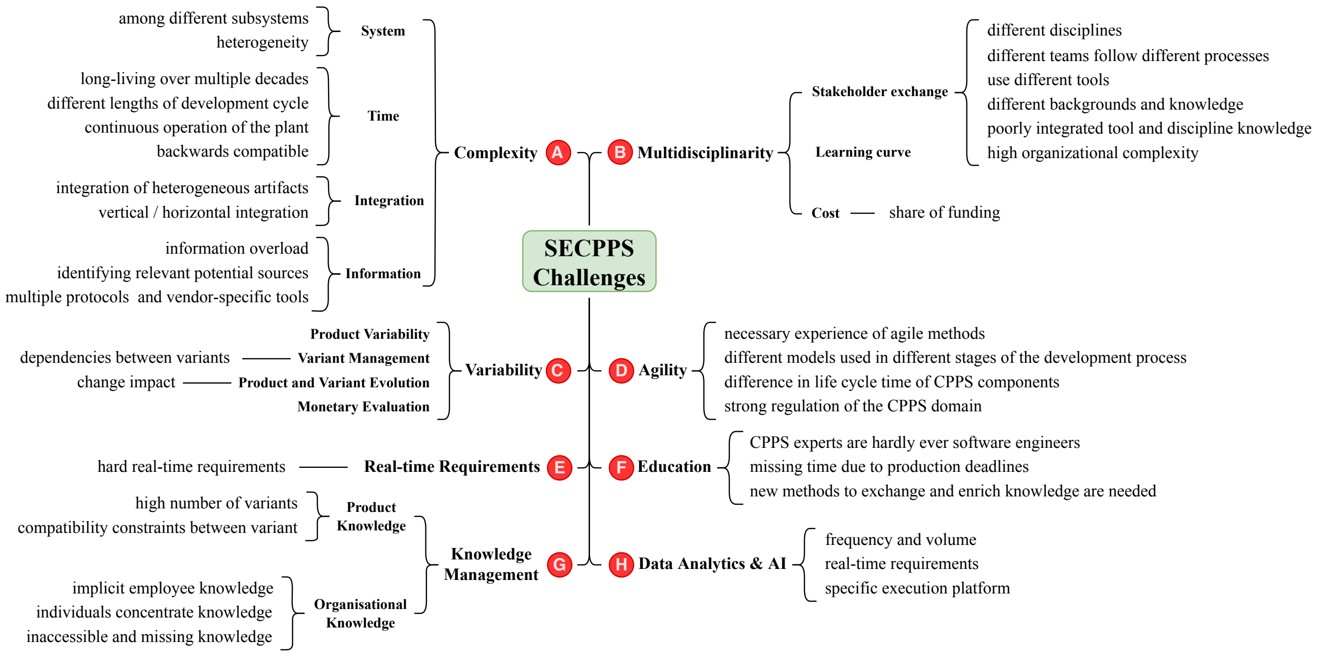
among different subsystems
heterogeneity
System
long-living over multiple decades
different lengths of development cycle
continuous operation of the plant
backwards compatible
Time
integration of heterogeneous artifacts
vertical / horizontal integration
Integration
information overload
identifying relevant potential sources
multiple protocols and vendor-specific tools
Information
Complexity A
B Multidisciplinarity
Stakeholder exchange
different disciplines
different teams follow different processes
use different tools
different backgrounds and knowledge
poorly integrated tool and discipline knowledge
high organizational complexity
Learning curve
Cost
share of funding
SECPPS Challenges
Product Variability
dependencies between variants
Variant Management
change impact
Product and Variant Evolution
Monetary Evaluation
Variability C
D Agility
necessary experience of agile methods
different models used in different stages of the development process
difference in life cycle time of CPPS components
strong regulation of the CPPS domain
hard real-time requirements
Real-time Requirements E
F Education
CPPS experts are hardly ever software engineers
missing time due to production deadlines
new methods to exchange and enrich knowledge are needed
high number of variants
compatibility constraints between variant
Product Knowledge
implicit employee knowledge
individuals concentrate knowledge
inaccessible and missing knowledge
Organisational Knowledge
Knowledge Management G
H Data Analytics & AI
frequency and volume
real-time requirements
specific execution platform

Fig. 1. Overview of identified Software Engineering in Cyber-Physical Production Systems (SECPPS) challenges.

achieve a holistic view of the system **integrating these—often heterogeneous—artifacts**. One paradigm to handle complexity is model-driven systems engineering that employs models for describing aspects of a CPPS [14]. Today a slew of modeling languages is already used to model different aspects of CPPSs. Furthermore, standardized (IEC) languages, e.g., IEC 61131-3 [15] or IEC 61499 [16], exist for programming PLCs. However, integrating these artifacts entails **vertical integration**, i.e., across multiple abstraction or production layers, e.g., component, subsystem, system, and **horizontal integration**, i.e., across systems on the same layer. These integration challenges extend to tools, software, communication interfaces, and other components involved in the composition of *System-of-Systems*.

Information complexity. CPPSs produce large quantities of data, which can result in **information overload**. As reported by industry representatives, an example is a high number of event reports in human-machine interfaces (HMI). In addition, the data may differ in transmission velocity, quality, and validity, depending on its source and context. These aspects are directly related to the organization using CPPSs, in the most complex case distributed globally [17]. Due to the amount and time constraints of CPPS data, scalability of solutions becomes an issue [18]. In large quantities of data, **identifying potential relevant sources can be challenging**. As an example reported by industry, identifying sources for errors is not trivial when a product defect is recognized in quality control. Another aspect raising the complexity of information transmitted in CPPSs are the **multitude of protocols**, including their semantics, and **vendor-specific tools** to manage these protocols. In this regard, further standardization is needed [19].

B. Multidisciplinary

Various disciplines are involved in engineering and maintaining CPPSs. Software engineers responsible for developing, e.g., the control software for CPPSs [6] require inputs and knowledge from their upstream disciplines. In practice, as several industry participants in our workshop confirmed, **different discipline** teams follow a **variety of processes**, use **diverse tools**, and have **different backgrounds and knowledge**. The **tools and the discipline knowledge are poorly integrated** resulting in a **high organizational complexity** (cf. Section III-A). For the integration of models, there are approaches such as *mega models* [20] or approaches in the context of *Model-based Systems Engineering* that rely on model transformations [21], [22]. However, tool support and acceptance in industrial projects are still not sufficient. One approach often followed in practice is the definition of Minimum Viable Products (MVPs). Nevertheless, in a multidisciplinary context it is often unclear what can be understood as an MVP that satisfies the involved disciplines. As several industry participants of the workshop confirmed, for the developers of (control) software in the context of CPPSs, it remains challenging to consider the knowledge from multiple involved disciplines when building solutions.

Stakeholder exchange. In practice, important system features have to be defined across multiple disciplines in CPPS engineering. Therefore, representatives of the responsible disciplines should meet and interact regularly. Industry participants, however, also confirmed that, in practice, such meetings are rare and sometimes only occur once a year. Interaction is typically based on the exchange of documents and spreadsheets, some manually created, others exported from different tools. Based on the knowledge contained in these documents, software engineers develop and maintain the CPPS software, which requires a lot of experience.

Learning curve. One industry participant explained that it can take several years to get a software engineer in their domain up to speed. Many industries use legacy industry standards instead of modern software engineering approaches and tools, at least compared to software development/IT companies. As a result, it is hard to find software engineers who want to work in such multidisciplinary environments. Researchers in industry and academia are working on bringing modern approaches, such as DevOps, to industry [23]. However, in practice applying new approaches may lead to problems. For instance a lack of testing may cause integration issues, even in traditional software engineering [24].

Cost. According to a workshop participant from the plant building domain, a challenge for software engineering for CPPS is **the share of funding** compared to an overall CPPS project. In particular, this concerns the basic automation, which accounts to 0.5 to 2 percent of some projects' budget. A possible contribution of academia could be to emphasize the value of proper software in the long run.

C. Variability

CPPSs are typically developed and maintained over many years, resulting in a plethora of variants created to address specific customer requirements. As confirmed by all participants, **customers expect a system** (e.g., plant or machine) **customized to their specific needs**, especially in industries that typically deal with highly configurable products. **Dealing with variability in industry**, however, currently **depends too much on mostly tacit domain expert knowledge and custom-built tools** focusing on particular artifacts and software and hardware platforms [5], [6].

Product Variability. Two industry participants, one from special vehicle construction and one from packaging management, mentioned the challenge of the desired **high product variability in modern production**, a.k.a lot-size 1 and configure to order. While the first participant noted that their products are highly individual, the second added that specific business processes to model them are still missing. These mentions are confirmed in recent literature [25], [26] and recent approaches try to **consider variability from multiple, i.e., product, process, and business perspectives** [27], [28]. Industry 4.0 aims to facilitate such flexibility and adaptability to address product variability to meet customer requirements [29], [30].

Variant Management. As multiple industry participants of our workshop pointed out, it is particularly challenging to manage the high number of variants and especially their (potential) compatibility. The **dependencies between variants** need to be modeled for this purpose. For instance, a large European truck company (not represented at our workshop) tries to address this issue by large constraint databases, which have been presented during a keynote by Mattias Nyberg [31] at the 25th ACM International Systems and Software Product Line Conference. This is something the Software Product Line (SPL) community has focused on for over three decades, resulting in a **slew of variability modeling approaches** [32]. Even though the challenges of integrating variability modeling in CPS engineering have been discussed [33] and several approaches have been developed to extract variability from various (industrial) artifacts [28], [34]–[36], **only few** of these approaches **seem to have reached industry**, especially regarding CPPS [8].

Product and Variant Evolution. The participants also reported product evolution and its **change impact** on the production system as an essential challenge (cf. also [37]). These challenges mean that the production systems and their software have to be re-configured to meet new requirements and manufacture new or changed products. However, the impact of product evolution should be predictable in advance. While SPL engineering can provide the means to model product variability, and there are works trying to improve artifact evolution [38], [39], SPL evolution is still challenging for software engineering [40], [41].

Monetary Evaluation. One industry stakeholder noted the monetary evaluation of variants in CPPS is a challenge. For instance, how can the value of an additional product variant be automatically evaluated and is a customer willing to pay the additional cost (for the CPPS)? In SPL, the monetary value of software variants and their contribution to the overall return on investment of the product line is relatively negligible. However, in CPPS engineering the contribution of every single product type manufactured to the return on investment of the CPPS is crucial [36]. Current variability modeling approaches largely focus on features but do not consider a fixed set of products that should be produced or the monetary values of features. As far as costs are modeled, e.g., in the field of reuse economics [42], these **cost models do not take the issue of physical production processes and their variability into account**, which is a major issue for CPPS.

D. Agility

Agility for CPPSs can be regarded as continuous architecting, assessment, and twinning based on accepted reference architectures like the Reference Architecture Model Industry 4.0 (RAMI 4.0), standardization, simulation of (embedded) architectures for assessment as well as operation, maintenance, and evolution of Digital Twins (DTs) [43]. The main challenges in this regard can be subdivided into three main themes.

Experience. **Experience** with agile methods in engineering companies is crucial. However, according to one workshop stakeholder from the CPPS solution provider domain, it is **often lacking**, especially with **model-driven engineering methods**.

Verification. Different models are used in different stages of the development process, which must be synchronized when changes are applied. Verification and validation of the models and their contents become essential due to the complexity and variety of CPPS. However, **verification and validation** are not yet prominent in the academic literature about CPPS [17].

Different life-cycle times. While product life cycles tend to get gradually shorter [1], software update cycles are in the scope of weeks or months in agile environments, and production equipment is often used for decades (cf. Section III-A). The agile processes in software engineering for CPPS need to take these different scales into account. Nevertheless, **agility requires short innovation cycles** in CPPS engineering. However, according to a participant of the workshop, these innovation cycles are **often impeded by the strong regulation of the CPPS domain**.

E. Real-time Requirements

One workshop participant reported on the challenges induced by the presence of real-time requirements for the OT subsystems in a CPPS. OT subsystems in a CPPS must fulfill not only their functional requirements but they must also adhere to **hard real-time requirements**, such as response time under 10ms. Particularly, the heterogeneity of the OT subsystems (cf. Section III-A) impose different classes of real-time requirements. There are **various classes of applications in CPPSs interacting with each other**, where each of these classes imposes differently tight real-time requirements on the applied networks and on the software together with the PLC it is deployed to [11], [44]. Consequently, safety standards like the IEC 61508 suggest certain safety measures to ensure that the OT subsystems fulfill their real-time requirements.

Additionally, in modern CPSs the real-time-critical OT subsystems have to interact via fog computing with non– or soft-real-time and often cloud-based IT subsystems (cf. Section III-A), thereby requiring independence as well as synchronization between both technology areas [11], [44]. Finally, the advent of data analytics in the context of CPPSs impose further challenges regarding real-time requirements and computing capabilities (cf. Section III-H).

F. Education

Formal education. CPPS engineering combines several disciplines, which requires a range of formally educated experts in these areas. While students can often include subjects taught in other disciplines as electives, programs like mechatronics and software engineering contain multidisciplinary elements by design [45]. As the subject of mechatronics evolved from combining computer science, electronics, and mechanical engineering to form a topic of its own [46], a similar evolution seems beneficial for CPPSs. This way, knowledge from the electronics, computer science, and mechanics domain can be integrated with methods and tools from software engineering (cf. learning curve in Section III-B).

Job training. CPPSs are developed and maintained by professionals from a variety of fields. Early systems were often developed by electronics professionals who learned how to program, as indicated by representatives from the industry. Such systems may become challenging to maintain, as languages like C or Ladder Diagram are replaced by higher-level languages, such as Python and JavaScript, which are often oriented towards web services or cloud development. Even though most professionals working with CPPSs today learned to program as part of their formal education, they are **hardly ever software engineers**. The most significant problem in education on the job, as reported in the workshop, is **missing time** due to project deadlines, especially for experienced colleagues. In this regard, **new methods to exchange and enrich knowledge** are needed while partaking in productive processes.

G. Knowledge Management

CPPSs need to integrate domain-specific knowledge from heterogeneous information sources to provide the basis for, e.g., analytics, simulations and decisions (cf. tool integration in Section III-B). This knowledge is spread over the entire CPPS ecosystem, including its context.

Product knowledge. The trend toward mass customization leads to **products with a high number of variants** specified in product design (cf. Section III-C), with a high number of relationships and **compatibility constraints between different product variants**, and to highly specified production processes about which knowledge is distributed among human workers, supporting (information) systems, and the CPPS.

Organizational knowledge. Moreover, organizational knowledge about these products and processes as **implicit employee knowledge** is often **not made explicit**. This is especially challenging because **individuals concentrate knowledge** about a system over time, but it depends on those individuals to make the knowledge transfer work (cf. stakeholder exchange in Section III-B). Knowledge management systems [47] allow to combine these human-machine information sources and support knowledge generation and utilization. We can apply methods from requirements engineering to explicate knowledge from humans and identify **inaccessible and missing knowledge**. We need to apply methods to structure externalized knowledge, which includes contextual data-traces and meta-data [48] as well as knowledge in engineering and process models [49]. This knowledge can be organized in organizational knowledge bases [50] to enable the support of evolution and learning.

H. Data Analytics and AI

An industry partner from an internationally distributed automation company did not participate in the actual workshop. Based on a post-workshop interview with this person, we identified challenges induced from the field of data analytics for CPPSs, which was initially not considered in the workshop.

Data analytics capabilities in CPPSs include complex event processing, soft-real-time stream processing as well as Artificial Intelligence (AI). Thus, engineering challenges for data analytics systems, e.g., as discussed in [51], also apply to CPPSs. Some core challenges are similar, e.g., intrinsic complexity, interdisciplinarity, the need for agile development, or (edge-)distributed data processing in (soft-)real-time settings. Functional requirements for data analyses often emerge late, e.g., during experimentation or realization. Furthermore, data privacy and data consistency and correctness require special approaches, e.g., during quality assurance. The embedded nature of the OT subsystems of a CPPS aggravate existing challenges, e.g., **frequency and volume of machine data streams at high sampling rates**, hard **real-time requirements** (cf. Section III-E) or **specific execution platforms**, e.g., PLCs that require IEC languages for programming. It is worth mentioning that edge devices bridge both worlds and integrate a hard real-time IEC environment for OT with a separated IT environment, where data analytics, AI, and even containers for virtualization can be executed.

Nowadays, AI is frequently used in CPPS for condition monitoring or predictive maintenance, as confirmed by a recent survey in the IIP-Ecosphere project [52]. However, systematic development in industry settings impose specific demands, e.g., compliance with certain standards. Example challenges for ISO-13374 are the need for integrated application design tools or the compliant evolution and change management of existing solutions [53]. These topics will become more urgent as other use cases are explored, e.g., energy optimization, and the number of cases per company will grow.

IV. Lessons Learned and Research Agenda

Complexity is a multi-dimensional problem. System complexity, time complexity, integration complexity, and information complexity have to be considered in the CPPS context. Modeling techniques and domain-specific languages can help alleviate some of the complexity. However, they come at the cost of, e.g., additional maintenance effort to keep the models and the implementation consistent and require training of staff.

CPPSs require real multidisciplinary approaches and tools. In practice, tools and artifacts are often specific to the disciplines and hard to integrate. Thus, interchanging information via common formats, such as AutomationML, can only be the first step. Also, despite tooling, proper (cross-disciplinary) stakeholder interaction is hard to realize in practice [1], [54].

Product variability management for different goals influences development and business processes. Product management mainly focuses on product production and whether the implementing CPPSs produces it in batch or as single products, i.e., lot-size 1. Therefore, the resulting product variability must be expressed in and across the domains and development areas, covering mechatronic, software, and business aspects. Each area uses heterogeneous methods and tools, which makes it hard to combine various artifacts, and there are no scalable or maintainable one-size-fits-all approaches available [55], [56]. Therefore, there is a need for an integrated, yet multi-view approach, involving different perspectives of CPPS engineering and supporting different goals during development.

Current life-cycle management tools and tool integrations are not sufficient. Given the integration complexity, multidisciplinarity, and variability in the CPPS domain, the state-of-the-art tooling regarding life-cycle management and tool integration is insufficient in several ways. On the one hand, product lifecycle management (PLM) tools originate from the mechanical engineering discipline and are well-suited for managing models and data on physical parts. However, they abstract away too many aspects from software engineering artifacts. On the other hand, application lifecycle management (ALM) tools originate from the software engineering discipline and provide an integrated management environment amenable to software engineers but are not well-suited to integrate the artifacts of other (CPPSs) engineering disciplines. Both lifecycle management tool directions are slowly melting together (particularly through mutual buyouts of the corresponding tool vendors), but the CPPS domain is still far away from an adequate solution.

Approaches from academia often never reach industry. For example, a slew of variability modeling approaches has been proposed that were never actually tried out for industrial use cases [57]. While it could be worthwhile to investigate this in more detail for CPPSs, it is hard to get scientific credit for such application-oriented research, especially in the software engineering research community [58].

CPPS Variability, especially the maintenance and evolution of CPPS variants, will remain a problem. Only proper variability management tool support and automation of tedious maintenance tasks (keyword: round-trip engineering and product line evolution) in general could help, but existing approaches need to be adapted to the CPPS context.

Engaging human factors is crucial. In addition to software and hardware, the engagement of human factors is crucial. Engineers constitute a significant source of information about variants of products, variability constraints between different product variants, the CPPS and its processes. Exploring and modeling their knowledge, leveraging it in software, retaining knowledgeable employees, and providing them programming skill updates will continue to be essential factors.

Research agenda. To tackle the discussed challenges, we propose the following research agenda.

The goal must be to **allow disciplines to keep their local tools, artifacts, and ways of working, while still providing means to integrate knowledge**. To also support evolution, especially required for long-living CPPSs, **support for managing variability in space (variants) and time (versions)** is essential. While in software engineering, the latter aspect is often well-covered by version control systems, such as GIT, the former is often not well-integrated with these systems. Furthermore, some disciplines in CPPSs will not be able to work easily with such systems as they have mainly been designed for text-based artifacts in software engineering rather than, e.g., (graphical) engineering models. Thus, **existing variant and version management systems need to be adapted to the CPPS context**, e.g., based on ontologies. This is far from trivial, and a one-size-fits-all approach will not be possible. A good starting point, however, will be best practices from software engineering, such as using traceability links with GIT, PLM, and ALM tools.

An underlying issue of many challenges is that the disciplines in CPPSs do not *speak the same language*, due to different backgrounds, diverse tools, and various artifacts. Still, developing software for CPPSs requires knowledge about all these disciplines. Thus, one solution is to **define ontolo-**

gies that represents concepts from different disciplines covering the required knowledge and its meta-information. Another solution is to use **connected, consistency-supporting modeling approaches** applying **domain-specific modeling languages** and reusable model libraries as a part of **integrated knowledge bases**. These concepts need to be integrated towards a common understanding using methods, such as the Common Concept Glossary (CCG) approach [59], and allow for mappings or transformations between concepts.

Moreover, a truly **multidisciplinary approach to lifecycle management and tool integration** would be much more discipline-integrating and thereby promising for coping with the challenges discussed in this paper than simply melting PLM and ALM tools together.

DTs represent another important concept for simulation, integration, testing and property prediction. DTs seem also promising to facilitate **backwards compatibility** and **continuous operation** when used for verification and validation. Even if the state-of-the-art covers already many facets [48], there are many open issues such as **horizontal and vertical integration of different subsystems**, consideration of user preferences and user rights, and consideration of time behaviour.

Software engineering research must demonstrate within **long-term lighthouse projects** that **agility and model-driven engineering methods** can be well applied in industrial practice. Thus, we need more field piloting of approaches and methods, e.g., model-based and SPL approaches, and industry partners who agree to a long-term, cross-company cooperation.

V. Conclusion

In this paper, we presented software engineering challenges in CPPS engineering motivated by industry voices and backed by recent literature. We elicited the challenges during the third instance of the SECPPS-WS[1], to which we invited industry partners to present their current work and challenges. Nine practitioners from eight different European companies, developing diverse CPPSs, participated. The challenges were complemented with findings of a reading club established to find recent related literature and possible (partial) solution candidates for particular challenges. We discussed the particular challenges and collected essential lessons learned to guide possible research directions.

We found that the CPPS domain not only bears challenges for the engineering of software but also for its operation. Key challenges for software engineering include *Complexity*, *Multidisciplinary* and *Variability*. Unsolved challenges range from *the need for development approaches and tools, which foster a real multidisciplinary environment* over *academic approaches that do not reach industry* to *the evolution of CPPSs variants/artifacts*. These challenges need more attention by software engineering researchers and practitioners.

To this end, we defined an initial **research agenda** describing the first steps towards addressing these challenges. Furthermore, we will continue the SECPPS-WS series to develop and discuss potential solutions together with academics and practitioners.

Acknowledgement

We thank all (industrial) participants of the SECPPS Workshop series for their input and feedback. The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612 and the German Federal Ministry of Education and Research (BMBF – 01IS19084D) as well as the German Federal Ministry of Economic Affairs and Climate Action (BMWK, IIP-Ecosphere – 01MK20006C, SofDCar – 19S21002L).

References

[1] S. Biffl, D. Gerhard, and A. Lüder, "Introduction to the multi-disciplinary engineering for cyber-physical production systems," in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 2017, pp. 1–24.

[2] L. Monostori, "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014.

[3] T. Bureŝ, D. Weyns, B. Schmerl, J. Fitzgerald, A. Aniculaesei, C. Berger, J. a. Cambeiro, J. Carlson, S. A. Chowdhury, M. Daun, N. Li, M. Markthaler, C. Menghi, B. Penzenstadler, A. Pettit, R. Pettit, L. Sabatucci, C. Tranoris, H. Vangheluwe, S. Voss, and E. Zavala, "Software engineering for smart cyber-physical systems (sescps 2018) - workshop report."

[4] B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *Journal of Systems and Software*, vol. 110, pp. 54–84, 2015.

[5] M. Galster, U. Zdun, D. Weyns, R. Rabiser, B. Zhang, M. Goedicke, and G. Perrouin, "Variability and complexity in software design: Towards a research agenda," *SIGSOFT Softw. Eng. Notes*, vol. 41, 2017.

[6] R. Rabiser and A. Zoitl, "Towards mastering variability in software-intensive cyber-physical production systems," *Procedia Computer Science*, vol. 180, pp. 50–59, 2021, proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020).

[7] T. Bureŝ, D. Weyns, B. Schmer, E. Tovar, E. Boden, T. Gabor, I. Gerostathopoulos, P. Gupta, E. Kang, A. Knauss, P. Patel, A. Rashid, I. Ruchkin, R. Sukkerd, and C. Tsigkanos, "Software engineering for smart cyber-physical systems: Challenges and promising solutions," *SIGSOFT Softw. Eng. Notes*, vol. 42, no. 2, p. 19–24, jun 2017.

[8] T. Berger, J.-P. Steghöfer, T. Ziadi, J. Robin, and J. Martinez, "The state of adoption and the challenges of systematic variability management in industry," *Empirical Software Engineering*, vol. 25, no. 3, 2020.

[9] R. Rabiser, B. Vogel-Heuser, M. Wimmer, A. Wortmann, and A. Zoitl, "Workshop on Software Engineering in Cyber-Physical Production Systems (SECPPS), 2nd Edition," in *Software Engineering*, ser. LNI, vol. P-320. Gesellschaft für Informatik e.V., 2022, pp. 105–106.

[10] M. Felderer and G. H. Travassos, Eds., *Contemporary Empirical Methods in Software Engineering*. Springer, 2020.

[11] D. Serpanos, "The cyber-physical systems revolution," *Computer*, vol. 51, no. 3, pp. 70–73, 2018.

[12] I. Groher and R. Weinreich, "An interview study on sustainability concerns in software development projects," in *43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2017.

[13] F. Li, G. Bayrak, K. Kernschmidt, and B. Vogel-Heuser, "Specification of the requirements to support information technology-cycles in the machine and plant manufacturing industry," *IFAC Proceedings Volumes*, vol. 45, no. 6, pp. 1077–1082, May 2012.

[14] L. Berardinelli, A. Mazak, O. Alt, M. Wimmer, and G. Kappel, "Model-driven systems engineering: Principles and application in the CPPS domain," in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 2017, pp. 261–299.

[15] M. Tiegelkamp and K.-H. John, *IEC 61131-3: Programming industrial automation systems*. Springer, 2010.

[16] International Electrotechnical Commission, "IEC 61499-1, Function Blocks - part 1: Architecture: Edition 2.0," Geneva, 2012.

[17] A. Wortmann, O. Barais, B. Combemale, and M. Wimmer, "Modeling languages in Industry 4.0: an extended systematic mapping study," *Software and Systems Modeling*, vol. 19, no. 1, pp. 67–94, 2020.

[18] M. Brichni and W. Guedria, "Data analytics challenges in industry 4.0: A case-based approach," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2018.

[19] J. L. Romero-Gázquez and M. Bueno-Delgado, "Software architecture solution based on sdn for an industrial iot scenario," *Wireless Communications and Mobile Computing*, 2018.

[20] A. Seibel, S. Neumann, and H. Giese, "Dynamic hierarchical mega models: comprehensive traceability and its efficient maintenance," *Software and Systems Modeling*, vol. 9, no. 4, pp. 493–528, 2010.

[21] J. Holtmann, R. Bernijazov, M. Meyer, D. Schmelter, and C. Tschirner, "Integrated and iterative systems engineering and software requirements engineering for technical systems," *Journal of Software Evolution and Process*, vol. 28, no. 9, pp. 722–743, 2016.

[22] C. Heinzemann, O. Sudmann, W. Schäfer, and M. Tichy, "A discipline-spanning development process for self-adaptive mechatronic systems," in *Intl. Conf. on Software and Systems Process*. ACM, 2013, pp. 36–45.

[23] P. Brauner, M. Dalibor, M. Jarke, I. Kunze, I. Koren, G. Lakemeyer, M. Liebenberg, J. Michael, J. Pennekamp, C. Quix, B. Rumpe, W. van der Aalst, K. Wehrle, A. Wortmann, and M. Ziefle, "A computer science perspective on digital transformation in production," *ACM Trans. Internet Things*, vol. 3, no. 2, feb 2022.

[24] E. Laukkanen, J. Itkonen, and C. Lassenius, "Problems, causes and solutions when adopting continuous delivery—a systematic literature review," *Information and Software Technology*, vol. 82, pp. 55–79, 2017.

[25] X. L. Hoang and A. Fay, "A capability model for the adaptation of manufacturing systems," in *ETFA*. IEEE, 2019, pp. 1053–1060.

[26] J. Pfrommer, D. Stogl, K. Aleksandrov, V. Schubert, and B. Hein, "Modelling and orchestration of service-based manufacturing systems via skills," in *ETFA*. IEEE, 2014, pp. 1–4.

[27] H. S. Fadhlillah, K. Feichtinger, K. Meixner, L. Sonnleithner, R. Rabiser, and A. Zoitl, "Towards multidisciplinary delta-oriented variability management in cyber-physical production systems," in *Proc. of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems*, ser. VaMoS '22. ACM, 2022.

[28] K. Meixner, K. Feichtinger, R. Rabiser, and S. Biffl, "Efficient production process variability exploration," in *Proc. 16th International Working Conference on Variability Modelling of Software-Intensive Systems*. ACM, 2022.

[29] W. Motsch, K. Dorofeev, K. Gerber, S. Knoch, A. David, and M. Ruskowski, "Concept for modeling and usage of functionally described capabilities and skills," in *ETFA*. IEEE, 2021, pp. 1–8.

[30] P. I. 4.0, "Details of the Asset Administration Shell Part 1 (V. 3.0RC01)," German BMWI, Standard, Nov. 2020, https://bit.ly/37A002I.

[31] M. Nyberg. (2021, 09) Generating safety cases for large-scale industrial product lines. Keynote at 25th ACM International Systems and Software Product Line Conference. [Online]. Available: https://splc2021.net/program/keynotes

[32] M. Raatikainen, J. Tiihonen, and T. Männistö, "Software product lines and variability modeling: A tertiary study," *Journal of Systems and Software*, vol. 149, pp. 485–510, 2019.

[33] J. Krüger, S. Nielebock, S. Krieter, C. Diedrich, T. Leich, G. Saake, S. Zug, and F. Ortmeier, "Beyond software product lines: Variability modeling in cyber-physical systems," in *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A*, ser. SPLC '17. New York, NY, USA: ACM, 2017, p. 237–241.

[34] M. T. Valente, V. Borges, and L. Passos, "A semi-automatic approach for extracting software product lines," *IEEE Trans. Softw. Eng.*, 2012.

[35] N. H. Bakar, Z. M. Kasirun, and N. Salleh, "Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review," *Journal of Systems and Software*, vol. 106, pp. 132–149, 2015.

[36] K. Feichtinger, K. Meixner, R. Rabiser, and S. Biffl, "Variability transformation from industrial engineering artifacts: An example in the cyber-physical production systems domain," in *Proc. 24th ACM International Systems and Software Product Line Conference*. ACM, 2020.

[37] R. Messnarz, A. Much, C. Kreiner, M. Biro, and J. Gorner, "Need for the continuous evolution of systems engineering practices for modern vehicle engineering," in *European Conference on Software Process Improvement*. Springer, 2017, pp. 439–452.

[38] C. Seidl, F. Heidenreich, and U. Aßmann, "Co-evolution of models and feature mapping in software product lines," in *16th Int. Software Product Line Conference - Vol. 1*, ser. SPLC '12. ACM, 2012, p. 76–85.

[39] K. Feichtinger, D. Hinterreiter, L. Linsbauer, H. Prähofer, and P. Grünbacher, "Guiding feature model evolution by lifting code-level dependencies," *Journal of Computer Languages*, vol. 63, 2021.

[40] M. Laguna and Y. Crespo, "A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring," *Science of Computer Programming*, vol. 78, 2013.

[41] M. Marques, J. Simmonds, P. Rossel, and M. Bastarrica, "Software product line evolution: A systematic literature review," *Information and Software Technology*, vol. 105, pp. 190–208, JAN 2019.

[42] M. S. Ali, M. A. Babar, and K. Schmid, "A comparative survey of economic models for software product lines," in *35th Euromicro Conference on Software Engineering and Advanced Applications*, 2009.

[43] E. Y. Nakagawa, P. O. Antonino, F. Schnicke, T. Kuhn, and P. Liggesmeyer, "Continuous systems and software engineering for industry 4.0: A disruptive view," *Information and Software Technology*, 2021.

[44] W. A. Khan, L. Wisniewski, D. Lang, and J. Jasperneite, "Analysis of the requirements for offering industrie 4.0 applications as a cloud service," in *26th International Symposium on Industrial Electronics*. IEEE, 2017.

[45] S. Coşkun, Y. Kayıkcı, and E. Gençay, "Adapting engineering education to industry 4.0 vision," *Technologies*, vol. 7, no. 1, p. 10, 2019.

[46] M. Grimheden and M. Hanson, "Mechatronics—the evolution of an academic discipline in engineering education," *Mechatronics*, 2005.

[47] F. Ansari, "Knowledge management 4.0: Theoretical and practical considerations in cyber physical production systems," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1597–1602, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[48] F. Becker, P. Bibow, M. Dalibor, A. Gannouni, V. Hahn, C. Hopmann, M. Jarke, I. Koren, M. Kröger, J. Lipp, J. Maibaum, J. Michael, B. Rumpe, P. Sapel, N. Schäfer, G. J. Schmitz, G. Schuh, and A. Wortmann, "A Conceptual Model for Digital Shadows in Industry and its Application," in *Conceptual Modeling (ER'21)*. Springer, 2021.

[49] T. Brockhoff, M. Heithoff, I. Koren, J. Michael, J. Pfeiffer, B. Rumpe, M. S. Uysal, W. M. P. van der Aalst, and A. Wortmann, "Process Prediction with Digital Twins," in *Int. Conf. on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. ACM/IEEE, 2021.

[50] H. Panetto, B. Iung, D. Ivanov, G. Weichhart, and X. Wang, "Challenges for the cyber-physical manufacturing enterprises of the future," *Annual Reviews in Control*, vol. 47, pp. 200–213, 2019.

[51] O. Hummel, H. Eichelberger, A. Giloj, D. Werle, and K. Schmid, "A Collection of Software Engineering Challenges for Big Data System Development," in *Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2018, pp. 362–369.

[52] C. Niederee, H. Eichelberger, H.-D. Schmees, A. Broos, and P. Schreiber, "Ki in der produktion – quo vadis?" Oct. 2021, https://www.iip-ecosphere.de/wp-content/uploads/2021/11/IIP-Ecosphere-Whitepaper-zur-Umfrage-KI-in-der-Produktion.pdf. [Online]. Available: https://doi.org/10.5281/zenodo.6334521

[53] F. Pasic, "Model-driven development of condition monitoring software," in *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '18. ACM, 2018, p. 162–167.

[54] L. Waltersdorfer, F. Rinker, L. Kathrein, and S. Biffl, "Experiences with technical debt and management strategies in production systems engineering," in *Proceedings of the 3rd International Conference on Technical Debt*. New York, USA: ACM, jun 2020, pp. 41–50.

[55] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wąsowski, "Cool features and tough decisions: A comparison of variability modeling approaches," in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, ser. VaMoS '12. New York, NY, USA: ACM, 2012, p. 173–182.

[56] O. Oliinyk, K. Petersen, M. Schoelzke, M. Becker, and S. Schneickert, "Structuring automotive product lines and feature models: an exploratory study at opel," *Requirements Engineering*, vol. 22, pp. 105–135, 2017.

[57] R. Rabiser, K. Schmid, M. Becker, G. Botterweck, M. Galster, I. Groher, and D. Weyns, "A Study and Comparison of Industrial vs. Academic Software Product Line Research Published at SPLC," in *22nd International Systems and Software Product Line Conference (SPLC 2018)*. Gothenburg, Sweden: ACM, 2018, pp. 14–24.

[58] L. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh, "The case for context-driven software engineering research: generalizability is overrated," *IEEE Software*, vol. 34, no. 5, pp. 72–75, 2017.

[59] F. Rinker, L. Waltersdorfer, K. Meixner, and S. Biffl, "Towards Support of Global Views on Common Concepts employing Local Views," in *ETFA*. IEEE, 2019, pp. 1686–1689.