2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)

Exploring Virtual Reality as an Integrated Development Environment for Cyber-Physical Systems

Tommi Mikkonen
Department of Computer Science
University of Helsinki, Helsinki, Finland
tommi.mikkonen@helsinki.fi

Kai-Kristian Kemell
Department of Information Technology
University of Jyväskylä, Jyväskylä, Finland
kai-kristian.o.kemell@jyu.fi

Petri Kettunen
Department of Computer Science
University of Helsinki, Helsinki, Finland
petri.kettunen@helsinki.fi

Pekka Abrahamsson
Department of Information Technology
University of Jyväskylä, Jyväskylä, Finland
pekka.abrahamsson@jyu.fi

***Abstract*—Cyber Physical Systems (CPS) development approaches tend to start from the physical (hardware) perspective, and the software is the final element in the process. However, this approach is unfit for the more software-intensive world that is increasingly iterative, connected, and constantly online. Many constraints prevent the application of iterative, incremental, and agile development methodologies, which now are the norm for many other fields of software. Time-consuming system validation can only start when both hardware and software components are ready, which implies that the software delivery and quality is almost always the final bottleneck in the CPS development and integration. Also organizational issues raise concerns – CPS development teams are nowadays often geographically distributed, which can result in delays in the process, shortcomings, and even mistakes. In this paper, we propose using our envisioned open-source Virtual Reality-based Integrated software Development Environment (VRIDE) for developing the next generation, increasingly software-intensive CPSs in efficient ways.**

***Index Terms*—Virtual Reality (VR), Cyber-Physical Systems (CPS), Integrated Development Environment (IDE), Embedded Systems, Virtualization, Digital Twin, Virtual Twin, Simulation, Agile Software Development, Collaboration.**

I. Introduction

The Internet of Things (IoT) represents the next significant step in the evolution of the connectivity and the rise of machines around us [18]. In such cyber-physical systems (CPS), physical and software components are deeply intertwined, each operating on different spatial and temporal scales, exhibiting multiple and distinct behavioral modalities, and interacting with each other in a myriad of ways that change with context, in real time [15]. In the future, they will be ever more complex with many cross-cutting issues, such as cyber-security and usability. Furthermore, many CPSs are critical by their fundamental nature, like electricity systems such as Smart Grids [21]).

As a consequence of this development, everyday software-enabled things in our surroundings are becoming connected and programmable dynamically. Furthermore, the majority of CPS class systems will never sleep (e.g., Smart Grids), and the number of computing devices (cyber components) will also be significantly larger and deployed in much more dynamic and complex topologies than in traditional environments [18]. This in turn implies that the focus of CPS-related product development will inevitably shift from individual devices to software that powers and controls them both in local and in connected setting – in essence, Weiser's vision of the invisible computer [20] is finally becoming everyday reality. In all the role of software and the needs for its agile yet dependable development become thoroughgoing. To reach this goal, we need software engineering approaches that are better suited to CPS development than those we are commonly using now [4].

In this paper we propose using virtual reality and other virtualization techniques for building an IDE development environment for CPS systems of the programmable world. Furthermore, we provide first-hand experiences gained with the prototype implementation and evaluate its potential benefits for future CPS software development.

II. Background and Motivation

A. CPS Characteristics

Cyber-Physical Systems (CPS) are systems that simultaneously act in the physical and digital space, comprising both physical and computational processes [12]. Typical examples of CPSs include drones, various robots, and autonomous vehicles (e.g. [5]) and also larger, complex systems such as Smart Grids [21]. Since a major part in their development includes the design of physical, mechanical and electrical elements, the development has been executed under their terms, and software has traditionally been a final element to include in the system. Now, as software as a key enabling technology and analytics powered by software is becoming a major factor in innovation in CPSs [13], [14], the situation is changing radically, and software engineering is receiving attention already early on in the development.

Up until recently, CPSs have largely been confined into factories as factory robots and into other such closed environments, away from the public eye. Nowadays they are becoming more interconnected (IoX) and ubiquitous. With advances in AI and machine learning, CPSs have been able to become more reliably autonomous while also taking on more generic tasks. A factory robot that is programmed to do the exact same action in the exact same way is far from being as sophisticated as a CPS that acts in a reactive fashion and is able to perform a multitude of tasks under different, changing circumstances.

Today, CPSs are developed across industries. The automotive industry has often made the headlines with its autonomous vehicles ranging from cars to trucks. Drone-based systems are being developed for military, surveillance, and shipping (e.g. Amazon's Prime Air[1]) purposes. The aforementioned factory robots are becoming more sophisticated and taking on more demanding and changing tasks in smart factories as so-called cobots [19]. Moreover, with recent advances in Artificial Intelligence (AI) the increasing prominence of the IoT, CPSs are becoming common across various industries and fields of research [16].

As in addition to running digital software in the cyber space, CPSs operate in the physical space, their development differs from traditional software development of purely digital software. In developing CPSs, the physical manifestation of the system has to be both modelled digitally and ultimately constructed physically. Thus, in developing CPSs, a working digital simulation of the physical aspects of the system has to be constructed and employed for testing and simulation purposes. As a result, common software development environments alone are not enough for CPS development with various different engineering disciplines as they do not contain the tools necessary for digitally simulating and integrating the physical aspect of the systems. In addition to traditional software development environments, various modelling tools are used to model and simulate the physical aspect of the CPS for this reason. Typically, these simulations are made using digital twins, simulations that cover the macro geometrical and micro atomic dimensions of the physical systems [17]. Constructing simulations accurate enough to be called digital twins requires high levels of expertise and supporting dedicated development environments.

Aside from the complexity resulting from the need for simulation, CPSs tend to be complex systems with distributed software components. For example, an autonomous car could have software running on car ECUs (Engine Control Unit), a gateway-like hardware, edge computing, and cloud computing (connected cars). These different software components might need to be developed in different levels of abstraction, and using different programming languages. This makes CPS development challenging from the point of view of development environments as well. The need for digitally modelling and simulating the physical counterpart of the system during development also further adds to this complexity. Finally, various different hardware platforms and operating systems can also be involved in the development ranging from hard real-time kernels to large process automation systems.

Arguably, developing CPSs is challenging not only from the point of view of the multi-disciplinary expertise required (modelling, programming, engineering etc.), but also from the tooling perspective. Integrated tool chains for CPS development are currently still lacking. Modelling and programming need to generally be done using separate software. Solutions for modelling the software components (e.g. software stack, power estimation, network, or sensing of the environment) of generic CPSs are still lacking. Existing solutions are largely designed for either specific operating system or specific devices. However, the need for integrated development environments for CPSs have also been pointed out in e.g. [6]. Moreover, there are various activities that aim at integrated tool chains. However, even recent efforts such as COSSIM[2] and INTO-CPS[3] still lack tools for managing the complexity of the surrounding environment and interactions between various objects.

Taking into account the current and future nature and contexts of CPSs, agile development methods, practices and ways of working excelling in general software contexts [8] could be highly beneficial also in modern CPS development. However, the inherent developmental requirements and constraints of CPSs like discussed above must be accommodated. This is where we see great opportunities of applying modern VR.

B. Virtual Reality and CPSs

Virtual Environments. Mixing virtual and real-world domains as such is not new as such, and it has been applied in various contexts [3]. So far such systems have been proposed for various use, including the following exploration of the International Space Station using VR facilities, allowing participants do things like dock cargo capsules, conduct spacewalks, and perform mission-critical tasks[4]; composing software for IoT devices that lets the developers to create applications faster than is possible when using directly the real hardware[5]; and analog modeling synthesizers that generates the sounds of traditional analog synthesizers using DSP components and software algorithms by simulating the behavior of the original electric and electronic circuitry to digitally replicate their tone[6]. Overall, various mixed reality (MR) applications are increasingly experimented and utilized in many traditional engineering areas such as factory plant design and building construction. All those involve more and more software-based and software-enabled functionalities.

Digital and Virtual Twins. In the context of virtual environment, the concept of digital twin refers to a digital replica of a physical system, used for reasoning about its physical

[1]https://www.amazon.com/Amazon-Prime-Air/

[2]http://www.cossim.org

[3]http://into-cps.org

[4]https://www.theverge.com/2017/3/9/14870462/oculus-nasa-esa-mission-iss-vr-space-station-simulator

[5]https://iotify.io/iotify-virtual-lab

[6]https://en.wikipedia.org/wiki/Analog_modeling_synthesizer

counterpart [9], [17]. In the context of CPSs, digital twins can be used in different ways across the life cycle of a product. Initially, they can be used to aid in designing the first physical form of the product and then be used to optimize it further [2]. Sensor data can be employed with digital twins to accurately simulate potential alterations to the physical system before any actual changes to it are made. The above is especially well-suited for CPSs that are costly or otherwise difficult to alter after their initial implementation. Being digital, digital twins can also be used regardless of physical location, making them attractive for geographically distributed teams. Furthermore, digital twins are often employed when developing CPSs that carry out more complex tasks in the physical world such as autonomous vehicles. To this end, expensive and safety-critical systems such as autonomous vehicles can benefit greatly from digital twins as they can be used to both teach the artificial intelligence in the safety of a digital environment as well as simulate e.g. disaster scenarios without incurring any material costs. By using digital twins, outcomes can be predicted both by using historical data and by simulating novel and atypical operation circumstances. In addition investigating failures and anomalies in live systems can be supported by replicating them in the twin part with the real system data.

Virtual Twins are a sub-set of digital twins, a technique used in various contexts [1]. Virtual twins, as their name implies, are digital twins with virtual presentations designed to be visually observed human actors. Through their visual presentation, virtual twins are intended to offer more information for human observers than ordinary digital twins.

The above is beneficial in various ways, which include gaining a better understanding of simulation results and generally seeing the CPSs in action in a virtual environment. Presently, virtual twins are especially underutilized in CPS development overall. They have been employed in the context of smart factories and industry 4.0, however, showcasing their potential relevance. In that context, virtual twins have been employed to e.g. train factory staff in the safety of a virtual environment, as well as to train robots for human interaction on the physical factory floor. In a similar fashion, virtual twins could be utilized in the development of CPSs in general.

III. Engineering the Solution: VRIDE

To tackle the challenges listed above, we propose using virtual reality based system as a development environment for CPSs. In contrast to moving in accordance to the departments of the developing organization, we wish to enable a more holistic approach, where the strict boundaries between the different engineering disciplines and development stages can be blurred in Agile ways. This calls for a new kind of a development environment where the tool chain forms an integrated experience.

Our prototype implementation VRIDE (Virtual Reality Integrated Development Environment) is an immersive development environment for CPS development in different domains (see Fig. 1[7]). The VRIDE vision has been designed to act as a demonstrator that satisfies the problems listed above:

- VR content for CPS cannot be produced inside Immersive Virtual Environments.
  - VRIDE supports modifications of software and other characteristics of the system inside the virtual environment.
- Physical prototypes are costly.
  - VRIDE enables building virtual versions of prototypes in digital form. Various levels of detail can be used, which enables early experimentation with less detail, and elaborated simulations when more detail is included.
- CPS development lacks an integrated tool chain.
  - VRIDE aims to provide a development tool chain for the virtual environment by introducing improved automation and interfacing between tools and simulations.
- Iterative, incremental, and agile development methodologies are challenging to employ in CPS development.
  - Using VR enables rapid feedback from the simulated environment.
- Collaboration between and inside teams in CPS development is difficult due to the highly heterogeneous tools.
  - The visual nature of VR makes it far easier for less tech savvy would-be users to provide feedback and participate in the design process.

Eventually, the goal is that the tool will support scripting, integration and automation to allow its connection to DevOps style pipelines. At present, the focus has been on developing the front-end and software development capabilities, with an ability to reflect the changes in software back to the virtual environment.

Fig. 2 shows a component diagram of the VRIDE system, given at a conceptual level to indicate the different functions. Technically, the system is built using the Unity engine[8], which is a commonly used system in games and other virtual environments, together with some auxiliary libraries. Full open source code of the demonstrator and associated documentation are made available at GitHub[9] for experiments and further research and development.

The prototype has given us some insights regarding how VRIDE is understood by various stakeholders. In the following, we list our view to VRIDE opportunities.

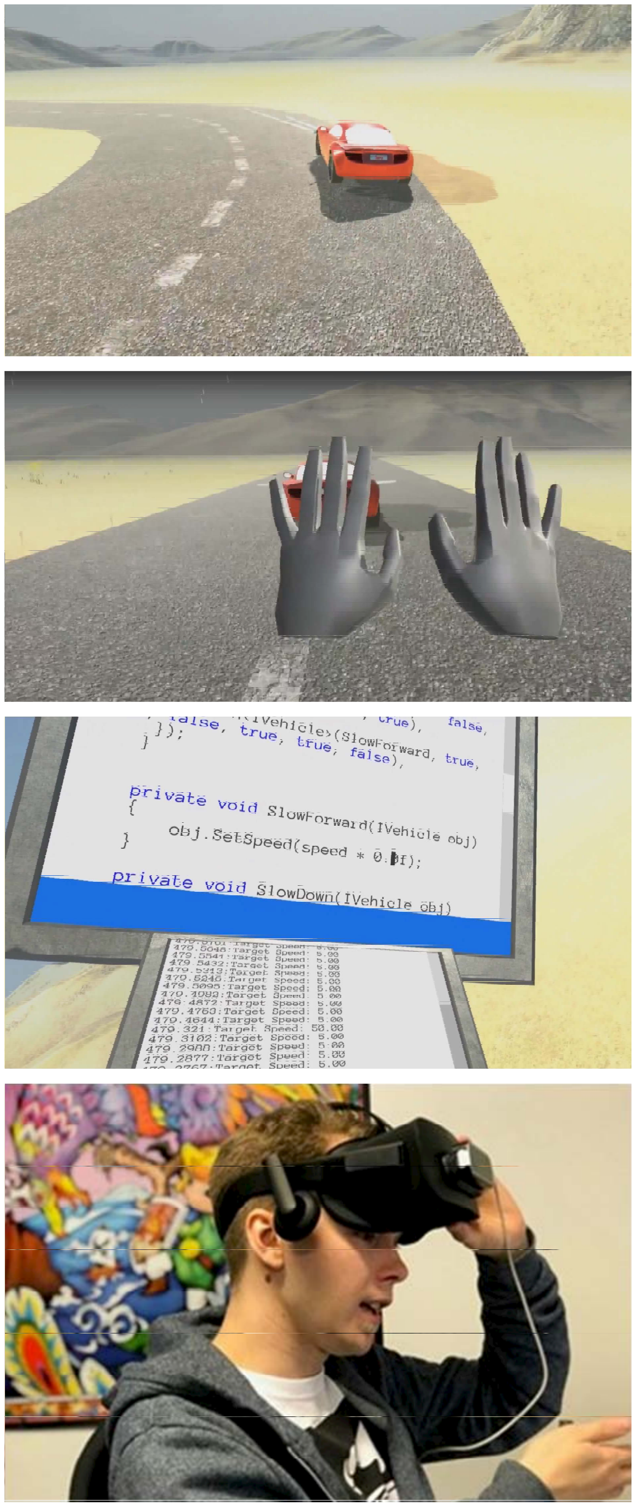
Enhancing systems thinking. Large complex CPSs may be extremely hard to comprehend. For instance modern electricity system Smart Grids are multilevel combinations of power network systems and distributed ICT systems. VRIDE could help software developers to realize the physical parts and their cyber connections. Moreover, the systems dynamics can be visualized with simulations and twins.

Supporting software requirements engineering. The software developers can examine the future use environment of

[7]Site http://bit.ly/vride-demo shows a demo video of current capabilities

[8]http://unity3d.com

[9]https://github.com/startuplabjyu/VRIDE



```
     false, true, true, false),
  });
}

private void SlowForward(IVehicle obj)
{
    obj.SetSpeed(speed * 0.0f);
}
private void SlowDown(IVehicle obj)
```

```
479.5648:Target Speed: 5.00
479.5541:Target Speed: 5.00
479.5432:Target Speed: 5.00
479.5313:Target Speed: 5.00
479.5246:Target Speed: 5.00
479.5095:Target Speed: 5.00
479.4982:Target Speed: 5.00
479.4872:Target Speed: 5.00
479.4763:Target Speed: 5.00
479.4644:Target Speed: 5.00
479.321:Target Speed: 50.00
479.3102:Target Speed: 5.00
479.2988:Target Speed: 5.00
479.2877:Target Speed: 5.00
```

Figure 1. Validating an Autonomous Vehicle in a Virtual Environment. From top to bottom: (i) A car running in a virtual environment; (ii) Programmer palms for editing the code and settings within the environment ; (iii) Access to vehicle code for altering its algorithms and behavior ; and (iv) A developer wearing a VR gear, needed to realistically view the 3D environment.

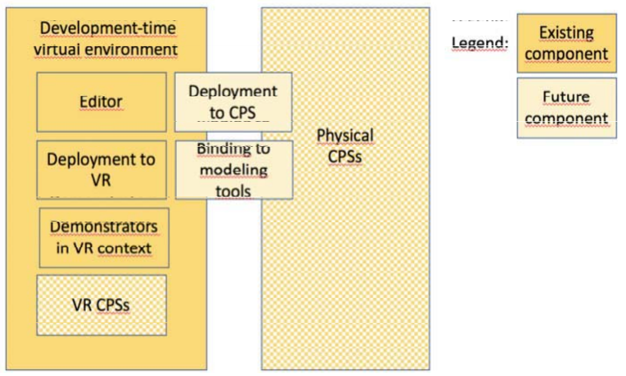


Figure 2. Parts of VRIDE we have implemented and foreseen extensions.

the CPS-under-development before its physical construction. This may assist in realizing the software requirements and intended behaviors. Furthermore, with VRIDE the software developers can revisit the VR "world" as often and whenever they need and want during the development. This is typically not possible for instance in case of running factories.

Capabilities for the future of CPS development. Overall, the role and share of software as a key enabling technology and basis for value creation in most CPSs are expected to increase in the future. Consequently, more software development work will be needed. More powerful software engineering capabilities are likely to be needed not only in pure software organizations but also in current non-software companies. Our VRIDE infrastructure aims to support such elaborate capabilities.

Shorter R&D feedback cycles. Virtual prototypes and designs can be used to gather feedback and evaluate the feasibility of a design faster especially in the early stage of the design process. Furthermore, even during production, digital twins of systems can be used to test intended modifications to the system without making any changes to the actual system.

More end-user involvement. The most notable strength of VR is its immersive, visual representation of a digital space. While involving end-users in the early phase of the product design process can be challenging, the visual nature of VR makes it far easier for less tech savvy would-be users to provide feedback and participate in the design process.

Interdisciplinary and distributed virtual collaboration. Effective CPS development requires multiple teams that contain expertise from various fields. The current state of the art, however, only offers 3D user interfaces designed for single users. Using such a system, teams could collaborate inside the same virtual environment, also regardless of geographical location. The collaboration can be used with technologies proposed in the context of collaborative online development environments, such as the Cored tool environment [11].

Supporting the adoption of newer development methodologies in CPS development. By providing an integrated tool chain for CPS development and thus enabling cyclic development, while also supporting end-user involvement and helping provide rapid feedback, VRIDE could help adopt top-of-the-line agile development methodologies such as DevOps and various Continuous SE methods.

Less context switching. By enabling developers to program and design VR content inside an Immersive Virtual Environment, less switching between environments is required, potentially speeding up development. Similar approaches have been proposed also in other settings, such as the Lively Kernel, where the developer can alter the behavior the system by using the framework itself [10].

Making hard-to-reproduce faults easier to investigate. While simulating accidents in the physical world is difficult and often impossible, virtual simulations can be run without incurring any material costs. Virtual simulations can be run time and time again to gather more data as well as to provide on-the-spot learning experiences.

Tool interoperability by providing an open source alternative. Presently, standardization between tools in the area is lacking. Commercial tools are not open source and are generally designed for highly specific use contexts or tailored to fit the context of each client. There is thus little interoperability between tools, making it difficult to make use of multiple tools to extend one another.

IV. Conclusions

Currently, CPS development lacks integrated tool chains. Consequently, the development is carried out using separate tools for programming, modelling and simulation. This results in problems with using modern agile software development methodologies that require automation and makes it more difficult for teams and team members to collaborate. In this paper, we propose using virtual reality environments in cyber-physical system development, following the ideas of previously built self-sustainable systems such as the Lively Kernel, where the Lively Kernel system can be used to modify itself [10].

Aside from solving this problem with the tool chain, the VRIDE virtual reality integrated development environment for CPS could benefit developers in various ways, including early and frequent testing of designs; collaboration with non-technical users and developers coming from other fields to collaborate alongside software developers in CPS development; and empower users by providing new opportunities such as remote collaboration and training inside IVEs.

Since the idea is novel and work reported in this paper is largely exploratory, there are several directions for future work. These include technical challenges associated with deep integration of all the modern development tools and infrastructure in the virtual reality, as well as usability and developer experience challenges, when one is using a virtual reality extensively extended periods of time. Furthermore, working in cooperation with industries where digital and virtual twins are an everyday tool would help us improve VRIDE and the key assumptions associated with it.

References

[1] M. Abramovici, J. C. Göbel, and P. Savarino. Virtual Twins as Integrative Components of Smart Products. In IFIP International Conference on Product Lifecycle Management, pp. 217-226. Springer, Cham, 2016.

[2] K.M. Alam and A. El Saddik. C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems. IEEE Access, 5, pp.2050-2062, 2017.

[3] Leif P. Berg, Judy M. Vance. Industry Use of Virtual Reality in Product Design and Manufacturing: A Survey. Virtual Reality (2017) 21:1–17

[4] T. Bures, D. Weyns, C. Berger, S. Biffl, M. Daun, T. Gabor, D. Garlan, I. Gerostathopoulos, C. Julien, F. Krikava, R. Mordinyi, and N. Pronios. Software Engineering for Smart Cyber-Physical Systems – Towards a Research Agenda: Report on the First International Workshop on Software Engineering for Smart CPS, ACM SIGSOFT Software Engineering Notes, 40(6), 28-32, ACM, 2015.

[5] X. Du, M. H. Ang, and D. Rus. Car Detection for Autonomous Vehicle: LIDAR and Vision Fusion Approach through Deep Learning Framework. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[6] J. El-Khoury, F. Asplund, M. Biehl, F. Loiret, and M. Törngren. A Roadmap Towards Integrated CPS Development Environments. In 1st Open EIT ICT Labs Workshop on Cyber-Physical Systems Engineering, May 2013.

[7] M. M. Fischer. Inception: A Creative Coding Environment for Virtual Reality, in Virtual Reality. In VRST '16 Proc. 22nd ACM Conference on Virtual Reality Software and Technology, pp. 339-340, 2016.

[8] B. Fitzgerald, K. Stol. Continuous Software Engineering and Beyond: Trends and Challenges. In RCoSE 2014 Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, pp. 1-9, 2014.

[9] E. Glaessgen and D. Stargel. The Digital Twin Paradigm for Future NASA and US Air Force Vehicles. In 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA, p. 1818. 2012.

[10] D. Ingalls, K. Palacz, S. Uhler, A. Taivalsaari, and T. Mikkonen. The Lively Kernel: A Self-Supporting System on a Web Page. In Self-Sustaining Systems, 31-50, Springer, Berlin, Heidelberg. 2008.

[11] J. Lautamäki, , A. Nieminen, J. Koskinen, T. Aho, T. Mikkonen, and M. Englund. CoRED: browser-based Collaborative Real-time Editor for Java web applications. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, pages 1307-1316, ACM, 2012.

[12] E. A. Lee. Cyber Physical Systems: Design Challenges. In Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on, 2008.

[13] J. Lee, H. A. Kao, and S. Yang. Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment. Procedia Cirp, 16, 3-8, 2014.

[14] M. Mikusz. Towards an understanding of cyber-physical systems as industrial software-product-service systems. Procedia CIRP, 16, 385-389, 2014.

[15] National Science Foundation. Cyber-Physical Systems. December 22, 2010.

[16] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In DAC '10 Proceedings of the 47th Design Automation Conference, pp. 731-736, 2010.

[17] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack. Shaping the Digital Twin for Design and Production Engineering. CIRP Annals, 66(1), pp. 141-144, 2017.

[18] A. Taivalsaari, and T. Mikkonen. A Roadmap to the Programmable World: Software Challenges in the IoT Era. IEEE Software, (1), 72-80, 2017.

[19] X. V. Wang, Z. Kemény, J. Váncza, and L. Wang. Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation. CIRP Annals, 66(1), pp. 5-8, 2017.

[20] M. Weiser. The Computer for the 21st Century. Scientific American, 265(3), pp. 94-104, 1991.

[21] Yu X, Xue Y (2016) Smart grids: a cyber–physical systems perspective. Proc. IEEE 104(5):1058–1070