Digital Twins for Continuous Deployment in Model-Based Systems Engineering of Cyber-Physical Systems

Joost Mertens[a], Joachim Denil[a]

[a]University of Antwerp, Groenenborgerlaan 171, 2020 Antwerpen, Belgium

Abstract

Cyber-Physical Systems (CPS) are required to operate over a longer lifetime. As such, their requirements can change, requiring updates to the system to be be rolled out continuously (Continuous Deployment) throughout the system's lifetime. The DevOps methodology provides a structured, quality assuring way to do so, as it integrates Development and Operations of a system in a continuous cycle. DevOps is generally applied in software development, however in the design of CPS, which follows a Model-Based Systems Engineering (MBSE) approach, it is not. This is because many challenges remain in the application of DevOps in MBSE. Our focus is on creating the foundations for continuous deployment of safety-critical CPS using digital twins of the CPS.k.

Keywords

Continuous Deployment, Cyber-Physical Systems, Model-Based Systems Engineering

1. Introduction

The design of future Cyber-Physical Systems (CPS), is faced with multiple challenges, as noted in [1, 2]. Because of the required increasing lifetime, systems are continuously improved, rather than the more traditional deliver and maintain lifecycle. Components can be modified, added, the scope of the system can be extended, the requirements changed, or the system repurposed. While the initial design of the system is done entirely at design time, future changes happen at runtime. The authors in [2] note that a flexible approach is needed for those CPS, and that models of the system are invaluable in creating such a flexible approach. Similarly, the authors in [1] argue that models of the system can be continuously updated as to follow the reality as closely as possible (a digital twin as a synchronized digital representation of the system), however, new methods are needed. Furthermore, Combemale and Wimmer [2] propose the DevOps methodology to integrate the design and operation cycle of a CPS. DevOps (Development and Operations) is a method that tightly integrates the design and operations of a system, reducing the time between the need for a change and the implementation of the change. However, they note that within the context of CPS, which are mostly designed in a Model-Based Systems Engineering (MBSE) approach, the use of models in DevOps is still faced with challenges: such challenges include but are not limited to the monitoring of the CPS in

QUATIC'2020: 8th Software Engineering Doctoral Symposium, Sept 08, 2020
EMAIL: joost.mertens@uantwerpen.be (J. Mertens); joachim.denil@uantwerpen.be (J. Denil)
ORCID: 0000-0002-8148-5024 (J. Mertens); 0000-0002-4926-6737 (J. Denil)


harsh conditions, reasoning about the redesign of the system, synchronization between the system in operation and its models, the virtual deployment of system changes in model-based testing of these heterogeneous systems.

Inspired by these challenges, our aim is to create the foundations for continuous deployment of safety-critical CPS using a digital twin. We aim to do this by applying the DevOps cycle for the MBSE of CPS. In relation to DevOps, we specifically tackle the challenges in the release, deploy and monitor phases, visually represented in green/dots in Figure 1. Additionally, we only tackle the challenges at the technical side of the engineering process, while challenges in the human aspect of the process are disregarded. In summary, our focus is the development of new digital-twin enabled methods and techniques to allow the continuous deployment of CPS. With these techniques, we aim to solve the challenges in the monitoring, release management and deployment of CPS.

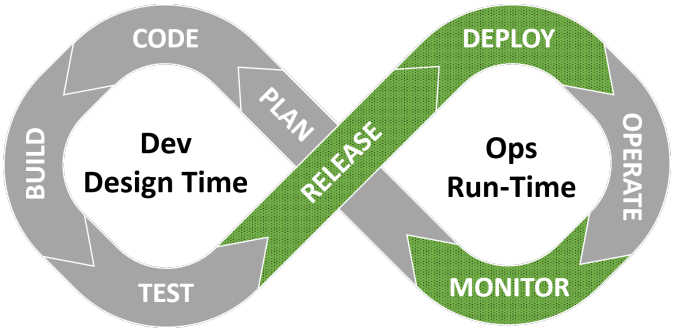


Figure 1: The DevOps cycle, with applicable stages marked in green/dots.

The remainder of this paper is structured as follows: in Section 2 we cover the state-of-the-art. Section 3 describes our objectives and approach, after which Section 4 lists past work and preliminary results. In Section 5, we look at the future, and lastly Section 6 concludes the paper.

2. State-of-the-Art

We find relevant research in the areas of (1) digital twins, (2) run-time monitoring in real-time systems and (3) continuous integration in MBSE.

2.1. Digital Twins

The digital twin is a popular concept in the industry 4.0 domain. In essence a digital twin is a virtual version of a physical entity, with which it shares data in both directions. As such, the application potential in the DevOps cycle is high, since it also exchanges information in such a way. In [3], the current state of research in digital twins is reviewed. Among their conclusions is the wide range of applications, and the adoption of digital twins by industry leaders. One of the noted applications is in System Design and Development. Strikingly is the fact that the list of current challenges as observed in research is still extensive, from scalability to trust/privacy issues. Furthermore, the authors note that within manufacturing, the use of

fully integrated digital twins is minimal. Besides Industry 4.0, there are ideas from other fields that show application potential for digital twins. One such idea is the data driven simulation paradigm, as demonstrated by Hu for wildfire simulation in [4]. In [4], the state of a wildfire simulation is continuously adjusted to match the real state of the wildfire based on sensor readings, showing strong similarities to digital twins.

2.2. Runtime Monitoring

Runtime monitoring in literature mainly covers the system monitoring itself but does not take the critical part of data-transfer into account for the DevOps cycle of CPS systems. Medhat et al. [5] investigated the run-time monitoring of CPS without a task model under timing and memory constraints. In their research, they propose a dynamic monitoring scheme based on control theory that monitors the system and tweaks itself based on multiple objectives. Similarly, in [6] models are instrumented statically for run-time monitoring under hard real-time constraints. Different data-transmission strategies for grouping and sending messages exist in literature e.g. grouping is possibly done statically or dynamically, at the local node or edge device, and other choices as seen in [7]. A CPS can also engage communication itself to notify peers and digital twins of state changes. In [8] the authors propose an architecture for such a case, where a filtering method makes sure that the system remains scalable.

2.3. Continuous Integration and Deployment in MBSE

In [9], the authors surveyed industrial companies for their adoption of continuous integration (CI) techniques in an MBSE setting. In summary, there is a lack of base functionality for CI in MBSE. Functional, non-functional, human and business difficulties where identified, some examples of which are: lack of tool interoperability, lack of synchronization between models, long build and test times, lack of willingness to model. Nonetheless, the paper shows that DevOps in the context of model-based (embedded) system design is possible, but that several challenges need to be resolved. In [10], the author proposes simulation as a solution to automate the difficult task of CI for embedded systems where dependencies exist between the software and the hardware platform. In [11], a study on continuous deployment for dependable systems concludes that the state of practice of agile development for safety critical systems is still immature. They also note that while the basic technologies to perform the continuous deployment do exist, some challenges remain, again on the topic of tool support and integration.

3. Research Objectives and Methodological Approach

3.1. Main Objective

The main goal is to define new methods that overcome current challenges concerning the continuous deployment of CPS, and tools that facilitate the integration of the digital twins in these methods. The underlying framework is the DevOps cycle for Model-Based Systems Engineering. Our approach is to apply multi-paradigm modeling principles to create such methods and techniques. Multi-paradigm modeling advocates the explicit modeling of all parts

of a system, at different abstraction levels using multiple formalisms [12]. At the center of our approach is a model based digital twin of the system as a shared common knowledge base, which allows further integration of design and operations.

3.2. Sub-Objectives

(i) Definition of a method to optimize data capture from a system in operation for the calibration and initialization of digital twins. To use a digital twin, there must be a synchronization between the real and physical world. This can be achieved by monitoring the system in operation and using the monitored data to initialize the digital twin. The data logging is done in the Monitor phase of the DevOps cycle. The problem at hand is a co-design problem. On the one hand, the initialization of a digital twin is a state estimation of the system, which is more accurate when more data is used. On the other hand, the CPS being monitored is a heterogeneous real-time system with limited resources. The system instrumentation introduces penalties in memory, execution time and, latency. As such, there is a trade-off between the accuracy of the initialization and the real-time behavior of the system. Additionally, the data must get to the digital twin. If the digital twin is physically separated from the system, network transmission is required. A network introduces additional limits and uncertainties: bandwidth, latency, packet loss.

(ii) Definition of a method to support release management and software deployment using the digital twin for viability checking. The goal is to create a method that can perform automated viability checking (testing) of a release on a product family and its undocumented variants. The challenge lies in the fact that within one product family, various system variants reside. These variants are either documented (variant by design, e.g. more expensive, faster) or undocumented (variant due to runtime changes, e.g. replaced parts, wear and tear, new environments that differ from the initial test conditions). In the testing phases of a piece of software, it can only be tested against the variants by design. Variants due to runtime changes should also be tested to ensure compatibility. The shared knowledge of the digital twin(s) of the system(s) implicitly contains the undocumented changes and can therefore exclude incompatible candidate systems through viability checking.

(iii) Exploration of computational architectures to support continuous deployment. The goal is to explore different architectures that enable the continuous deployment of system updates, specifically with regards to the release viability checking of (ii). Not all system architectures are efficient in this viability checking, especially with regards to the location of the digital twin. Examples of architectures could be: (a) running centralized digital twins and performing individual tests. (b) Running centralized digital twins, but grouping systems in compatible groups, and testing the group. (c) Running decentralized digital twins on edge nodes and performing individual test. These examples show the variability in the testing architecture. Finally, complex event processing can lump specific events together in the monitoring hierarchy.

4. Past Work and Preliminary Results

Our past and current work has mainly focused on sub-objective (i). As described in section 3.2, this objective concerns the co-optimization of a system and its digital twin to find the optimal trade-off between the instrumentation load on the system and the accuracy of the digital twin. For this trade-off analysis it is essential to have: (a) a method of verifying the instrumentation load on the system, and (b) a method to determine the sensitivity of parts of the model. Thus far, we have tackled the problem of verifying the load on the system in [13]. We did so through formal verification under the assumption of running a priority-preemptive scheduled Real-Time Operating System on our embedded system with only periodic tasks. Under the prescribed problem formulation, one obtains a non-linear problem, which we solve with a branch-and-bound solver. We have applied this method to two toy-examples, but also to the use-case of an autonomous radio-controlled 1/10th scale-car.

In the work on (a), we made various assumptions regarding the expected output from (b), which is only logical given the co-optimization approach. The focus of our current work is thus to replace those assumptions with factual data from developing (b). Currently, we are researching different types of data-assimilating digital twins of systems, of which we aim to generate sensitivity analyses as input for our method from (a). Then, the goal is to combine (a) and (b) in a single method.

5. Future Work and Expected Results

The objectives in section 3.2 are formulated in what we deem chronological order. After completing objective (i), we aim to first complete objective (ii), and then (iii), since objectives (ii) and (iii) are inherently linked. The goal for objective (ii) is to perform automated viability checking in the release and deploy stages of the DevOps cycle. This viability checking will be based on the current digital twin of the system, we envision an approach similar to virtual commissioning. In objective (iii) we then specifically aim to look at architectures supporting this release management. The reason being that we believe the needs for release management vary greatly between different CPS, e.g. the needs for an automated factory are different than those from a fleet of vehicles.

Lastly, we foresee that various assumptions will be made throughout this work, e.g. in our current partial solution to sub-objective (i), we still have assumptions with regards to the data-communication between the digital and physical world. After the first iteration through the objectives, we aim to perform a second development cycle, which will allow us to further elaborate on the assumptions made throughout the course.

6. Conclusions

In this paper we presented our views on using digital twins to enable continuous deployment in the field of model-based systems engineering for cyber-physical systems. We briefly introduced the challenges as covered in literature, and reviewed the related work most relevant to those

challenges we tackle. We described our main objective and three sub-objectives as well as our past, present and future work on these topics.

References

[1] S. Engel, D3.2 Policy Proposal “European Research Agenda for Cyber-physical Systems of Systems and their engineering needs", Technical Report, 2013.

[2] B. Combemale, M. Wimmer, Towards a Model-Based DevOps for Cyber-Physical Systems. Software Engineering Aspects of Continuous Development, Technical Report, 2019. URL: https://hal.inria.fr/hal-02407886.

[3] A. Fuller, Z. Fan, C. Day, C. Barlow, Digital Twin: Enabling Technology, Challenges and Open Research, 2019. arXiv:1911.01276.

[4] X. Hu, Dynamic data driven simulation, SCS M&S Magazine 5 (2011) 16–22.

[5] R. Medhat, B. Bonakdarpour, D. Kumar, S. Fischmeister, Runtime monitoring of cyber-physical systems under timing and memory constraints, ACM Transactions on Embedded Computing Systems 14 (2015). doi:10.1145/2744196.

[6] J. Denil, H. Kashif, P. Arafa, H. Vangheluwe, S. Fischmeister, Instrumentation and preservation of extra-functional properties of simulink models, in: Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, Society for Computer Simulation International, 2015, pp. 47–54.

[7] M. Nolan, M. J. McGrath, M. Spoczynski, D. Healy, Adaptive industrial IOT/CPS messaging strategies for improved edge compute utility, in: P. Pop, K.-E. Årzén (Eds.), Proceedings of the Workshop on Fog Computing and the IoT, IoT-Fog@IoTDI 2019, Montreal, QC, Canada, April 15, 2019, ACM, 2019, pp. 16–20. doi:10.1145/3313150.3313220.

[8] A. Berezovskyi, R. Inam, J. El-Khoury, M. Törngren, E. Fersman, R. Inam, Efficient State Update Exchange in a CPS Environment for Linked Data-based Digital Twins EEcient State Update Exchange in a CPS Environment for Linked Data-based Digital Twins, Technical Report, 2019.

[9] R. Jongeling, J. Carlson, A. Cicchetti, Impediments to Introducing Continuous Integration for Model-Based Development in Industry, in: Euromicro Conference on Software Engineering and Advanced Applications, 2019.

[10] J. Engblom, Continuous integration for embedded systems using simulation, in: Embedded World 2015 Congress, 2015.

[11] F. Warg, H. Blom, J. Borg, R. Johansson, Continuous Deployment for Dependable Systems with Continuous Assurance Cases, in: 2019 IEEE International Symposium on Software Reliability Engineering, WoSoCer workshop, IEEE Computer Society, 2019.

[12] H. Vangheluwe, J. Lara, P. Mosterman, An introduction to multi-paradigm modelling and simulation, Proceedings of the AIS’2002 Conference (2002).

[13] J. Mertens, M. Challenger, K. Vanherpen, J. Denil, Towards real-time cyber-physical systems instrumentation for creating digital twins, in: Proceedings of the 2020 Spring Simulation Conference, SpringSim ’20, Society for Computer Simulation International, San Diego, CA, USA, 2020.

A. Work Plan

A.1. Timeline

The work plan shown in Figure 2 is divided in 4 work packages. Three of these are for the sub-objectives, whereas the fourth foresees time for writing papers and general communication. Each year is divided in 4 quarters, each quarter is further split to 3 months. The People's months (PM) column gives an indication of the amount of hours each task receives. In the case of WP4, the timeline is colored light gray. This indicates that this is working time allotted throughout the entire 4 years. The work plan timeline cannot show the dependence of tasks on one-another, which is why the dependency relationship is provided by a design structure matrix.

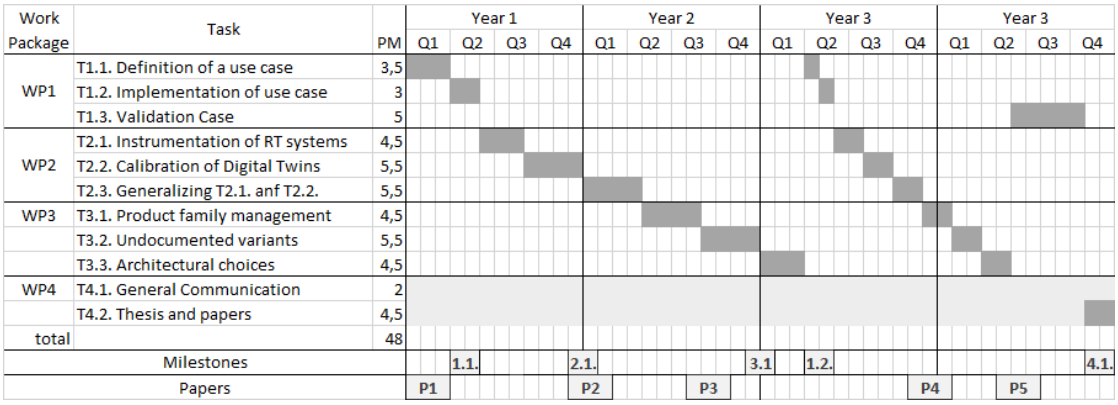


Figure 2: Timeline of the work plan indicating the time spent on each work package.

A.2. Design Structure Matrix

The design structure matrix shown in Figure 3 shows only one dependence problem, which is that the definition of the use case depends on the method being developed in T2.2., T3.1. and T3.2.. Depending on the abilities of this method, the use case must be defined, yet these abilities are not known beforehand. In our case we tackle this dependence by making assumptions on those abilities. By then iterating a second time through the design, as can be seen in the timeline, we can alter the implemented use case where necessary.

		T1.1.	T1.2.	T1.3.	T2.1	T2.2	T2.3	T3.1	T3.2	T3.3
Definition of a use case	T1.1.	T1.1.				X		X	X	
Implementation of use case	T1.2.	X	T1.2.							
Validation Case	T1.3.		X	T1.3.						
Instrumentation of RT systems	T2.1				T2.1					
Calibration of Digital Twins	T2.2		X		X	T2.2				
Generalizing T2.1. anf T2.2.	T2.3				X	X	T2.3			
Product family management	T3.1		X					T3.1		
Undocumented variants	T3.2		X					X	T3.2	
Architectural choices	T3.3							X	X	T3.3
rows = requiers input from										
columns = provides input to										

Figure 3: Design structure matrix indicating the dependencies between the tasks.

B. Poster

A visual poster summarizing the challenges discussed in this paper can be seen in Figure 4.

Figure 4: Poster summarizing the challenges in this paper.

Digital Twins for Continuous Deployment in Model-Based Systems Engineering of Cyber-Physical Systems

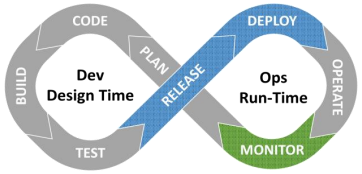
Joost Mertens, Joachim Denil; {joost.mertens, joachim.denil}@uantwerpen.be

CoSys-Lab Constrained Systems Lab University of Antwerp

University of Antwerp

https://www.uantwerpen.be/en/research-groups/cosys-lab/

Introduction:
Cyber-Physical Systems (CPS) are required to operate over a longer lifetime and system updates must be rolled out continuously [1 – 3]. DevOps provides a structured way to do so, but is not yet widely applied in Model-Based Systems Engineering (MBSE) due to many remaining challenges. We tackle some of these challenges by leveraging a digital twin of the system as common knowledge.



Representation of the DevOps cycle, with the applicable phases colored in blue and green.

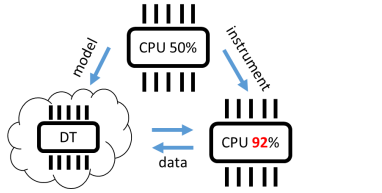
References

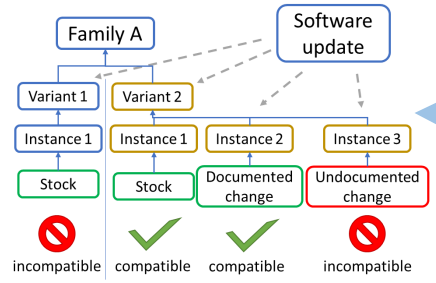
[1] Combemale, B., & Wimmer, M. (2019). *Towards a Model-Based DevOps for Cyber-Physical Sys-tems. Software Engineering Aspects of Continuous Development*. Retrieved from Springer-Verlag website: https://hal.inria.fr/hal-02407886

[2] Denil, J., Salay, R., Paredis, C., & Vangheluwe, H. (2017). Towards agile model-based systems engineering. *CEUR Workshop Proceedings*, *2019*, 424–429.

[3] Engel, S. (2013). D3.2 Policy Proposal "European Research Agenda for Cyber-physical Systems of Systems and their engineering needs". In *Towards a European Roadmap on Research and Innovation in Engineering and Management of CyberPhysical Systems of Systems*.

Challenge 1: capturing data for the digital twin of a constrained system. A digital twin as a shared knowledge base requires data to synchronize the digital twin with the real world.
Our objective: to define a method which optimally instruments the system, while respecting the hard-real time characteristics. This is a co-optimization between the digital twin and the physical system





Challenge 2: Managing software releases and deployment to product families. CPS are often grouped in product families with different (undocumented) variants in one family.
Our objective: to define a method that supports release management with compatibility testing based on the digital twin of the system. The digital twin identifies undocumented variants, and allows virtual deployment to ensure safety.

Challenge 3: variability in digital twin system architectures. Based on the type of systems, there may exist different amounts of digital twins. Thus, different architectures may be more suited for their execution.
Our objective: explore architectures that support the testing required to implement challenge 2. Additionally, explore how digital twins can represented a group of variants to reduce testing and computational overhead.