Continuous Integration Testing of Embedded Software with Digital Twin Prototypes

Alexander Barbie*†‡, Wilhelm Hasselbring‡, Niklas Pech*

*GEOMAR Helmholtz Centre for Ocean Research Kiel (Germany)
†Alfred Wegener Institute Helmholtz Centre for Polar and Marine Research (Germany)
‡Software Engineering Group, Christian-Albrechts-University, Kiel (Germany)

Abstract—Digital Twins may be employed for developing embedded software and can be classified in three subcategories by their level of integration with the corresponding physical objects/twins. We introduce and report on experience with a new category: the Digital Twin Prototype. Digital Twin Prototypes support engineers to test embedded software systems, without the need of a connection to a physical object. In CI/CD pipelines they can be used for integration testing and thus, allow for an agile verification and validation process. We developed and evaluated this approach to create an underwater network of ocean observation systems. The feasibility was shown in a demonstration mission in the Baltic Sea in October 2020.

Index Terms—Digital Twin Prototypes, Automated Testing, Integration Testing, Continuous Integration, Continuous Delivery, Embedded Software Systems

Introduction

For cyber physical systems and Industry 4.0 applications, the embedded software is an increasingly crucial asset that should be verified via automated tests. An example for Hardware-in-the-Loop (HiL) testing at large scale is Airbus with creating skeletons of their aircrafts in a test rig, containing the corresponding electrics, hydraulics and flight controls [1]. However, smaller companies cannot afford such redundant hardware just for the purpose of testing software.

A survey among $2,000$ decision makers about trends and challenges in software engineering found that quality is perceived in industry as the single most relevant condition to survive [2]. Yet, organizations struggle to achieve quality along with cost and efficiency [3]. Another study found that test automation is among the most popular topics for testing embedded software [4]. However, automatic quality assurance with continuous integration testing is a challenge in this context, since hardware is in the loop.

Digital twins may be employed on all layers in Industry 4.0 applications [5]. We leveraged the concept of digital twins for continuous integration testing of embedded software via the new approach of *Digital Twin Prototypes* (DTPs). DTPs can automatically be tested as proxies of the physical twins; thus, enabling automatic integration testing of embedded software. The digital thread between a physical twin and its digital counterpart is verified via appropriate integration tests with the DTP.

In the Helmholtz innovation project ARCHES (Autonomous Robotic Networks to Help Modern Societies) with a consortium of partners from AWI (Alfred-Wegener-Institute Helmholtz Centre for Polar and Marine Research), DLR (German Aerospace Center), KIT (Karlsruhe Institute of Technology), and the GEOMAR (Helmholtz Centre for Ocean Research Kiel), we developed several DTPs for ocean observation systems. The major aim of this project is to implement robotic sensing networks, which are able to autonomously respond to changes in the environment by adopting its measurement strategy, in both space and in the deep sea. A field report on employing DTPs in this context is published by Barbie et al. [6]. In the present paper, we introduce the general DTP approach for continuous integration testing of embedded software.

Digital Twins

> A digital twin (DT) is a digital model of a real entity, the physical twin (PT). It is both a digital shadow reflecting the status/operation of its physical twin, and a digital thread, recording the evolution of the physical twin over time [7].

Kritzinger et al. [8] classify three subcategories of digital twins by their level of integration with the physical twin, as illustrated in Figure 1:

- Figure 1a shows the digital model. There is no automated connection between the physical object and the digital model. All data exchange is done offline. State changes in the physical object do not immediately affect the digital model and vice versa.
- If there is an automated one-way data flow from the physical object to the digital object (see Figure 1b), then we refer to this as a digital shadow. A change in state of the physical object leads to a change of state in the digital shadow, but not vice versa.
- Figure 1c shows a fully integrated digital twin. The data flows are automated between the physical twin and the

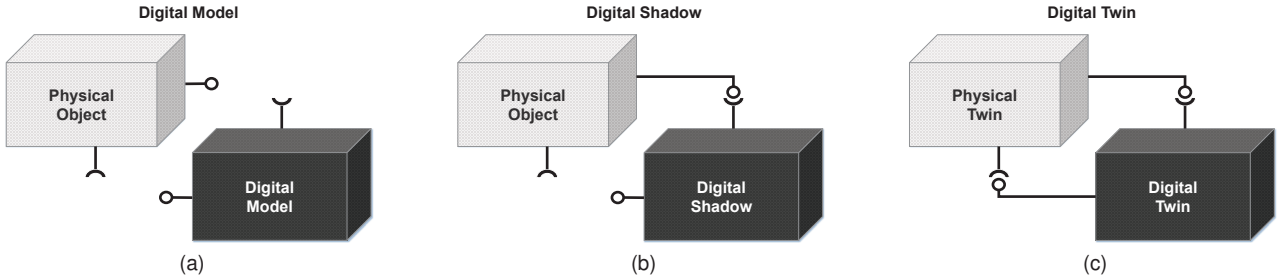


Fig. 1: Subcategories of digital twins by their level of integration with the Physical Twins, adapted from [8].

digital twin in both directions. In such a combination, the digital twin might also act as controlling instance of the physical twin. A change in state of the physical twin directly leads to a change in state of the digital twin and vice versa.

In the following, we introduce the Digital Twin Prototype as a fourth subcategory of a digital twin.

Digital Twin Prototypes

As shown in Figure 1, in the development of digital twins the physical object does not only play a role for the design of a DT, it is essential as source of the digital shadow that is synchronized to the DT. This reflects directly to a common embedded software engineering practice, where engineers still need to connect to a test rig and develop and test new code on the physical object. Consequently, developers are locally bound to that test rig and only a few developers can work on that test rig at the same time. This process is both laborious for teams with several (software) engineers and expensive.

> A Digital Twin Prototype (DTP) is the software prototype of a physical twin. The configurations are equal, yet the connected sensors/actuators are emulated. To simulate the behavior of the physical twin, the emulators use existing recordings of sensors and actuators. For continuous integration testing, the DTP can be connected to its corresponding digital twin, without the availability of the physical twin.

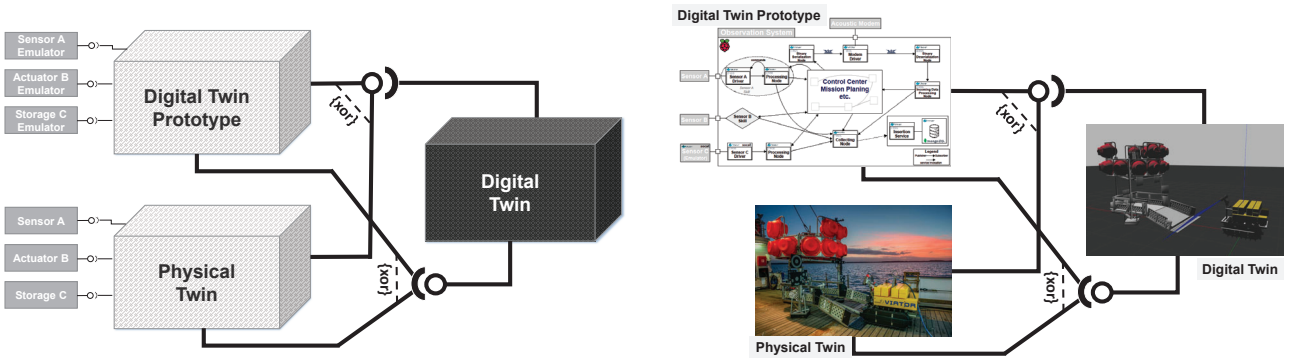
A DT does not need to integrate all the interfaces to real sensors and actuators, since all the data can be synchronized from its PT. Thus, the implementation of the software running the PT often differs from the software of the DT. We advocate an approach where the software of the PT and DTP is identical, and also part of the DT's software. Microservice architectures and containerization tools such as Docker make this possible. Furthermore, Docker allows a dependency management and a platform independent deployment.

The relationships between a PT, DT, and DTP in our approach are shown in Figure 2. The DTP has the same software configuration as the corresponding PT. Instead of real sensors/actuators the DTP uses emulated ones. The DT also uses the core software logic of the PT, yet without connected sensors and actuators, but with additional logic to control the PT and the DTP. To emulate interfaces other than Ethernet on the DTP, tools such as *socat* can be used to proxy an interface, e.g. a serial port, over TCP in Docker. Another way is to integrate simulation tools such as Gazebo, which we used to implement and test movement with caterpillar tracks. The important aspect is that we are able to integrate all interfaces into the development process and the inputs and outputs of all emulated hardware are similar to the real hardware. The emulators react to the same commands as their real counterpart, and they also return identically formatted data packages. Hence, a DTP is not only a software mock-up, it is able to replace the PT as the source of a digital shadow. Moreover, a well integrated DTP can be used to simulate *what-if* scenarios with more accurate results than a mere simulation.

The synchronization between the PT and DT can be solved via a microservice. The PT sends sensing/actuation data and state changes to the DT. The DT sends control commands to the PT. As a result, we get a replication of the core logic and the internal state of the PT as DT. This also supports engineers to localize the source of error, if the PT malfunctions. During the development process the DTP is connected to the DT, in production the PT is connected to the DT.

Consequently, engineers are able to develop new software modules in their local development environment and do not need the permanent connection to a physical test rig. The pressure to reduce costs [3] leads to many different approaches to switch from HiL to Software-in-the-Loop (SiL) development in the industry, mostly by using simulations tools. Bachuwar et al. [9] present a case study using ROS and different simulation tools to develop a realistic DT to virtually test autonomous vehicles. However, this approach does neither consider the connection between the PT and DT nor automated integration testing. Our DTP approach even considers the development of all aspects concerning the digital thread of a PT, e.g. applications to monitor and control PTs and cybersecurity aspects.

Note, this does not completely remove the factor hardware. Finally, the entire system still has to be tested on the PT before using it in production. Especially, performance tests can only be executed on the hardware used in production. Nevertheless, with DTPs we are able to test the software logic independent of the hardware.



(a) General concept design

(b) Realization and evaluation in ARCHES [6]

Fig. 2: Relationships of Digital Twin Prototypes with physical twins and digital twins

Continuous Twinning

By following continuous integration/continuous delivery (CI/CD) workflows the development of embedded software systems becomes an agile and incremental process. Starting with a prototype of a driver for a single piece of hardware, to entire production plants, to smart factories, the gap to agile software development closes. This does not only improve the software quality and shorten release cycles, it also allows additional stakeholders to participate in a feedback loop in the development process with the first software prototype. Adjusting software requirements, design flaws, or breaking up to complex processes can be fixed during development. With this method, digital twins evolve continuously in small incremental steps, rather than in major releases. Nakagawa et al. [10] envision and call this approach *Continuous Twinning*.

Developing a DTP of a physical object is a challenge, depending on the level of detail of all the components that are integrated into the system. However, the time that has to be invested into the development is worth the effort. A DTP reduces testing time in the further development process, since engineers are able to skip time-consuming tasks, such as preparing the test rig, connecting the development environment, starting the software, and executing scenarios manually. With DTPs this effort is reduced to the end of a release cycle.

Besides reducing the time that is needed for testing, does switching from HiL to SiL testing with DTPs reduce costs for redundant hardware and paves the way for more efficient development workflows that are difficult to implement for embedded software systems. DTPs become a key enabler for fully automated integration testing of embedded software systems in CI/CD pipelines. While building, testing, and releasing of software becomes possible for embedded software like in other fields of software engineering, integration testing with hardware interaction is expensive, due to the HiL testing, and is often done manually. Thus, the integration tests are a bottleneck in the verification and validation activities and hence, the release of new software. Nevertheless, with proper integration testing, developers increase the robustness of the embedded software systems. This may even embrace Industrial DevOps methods in the embedded field [11].

Field Experience

We developed and demonstrated the DTP approach in the context of the above-mentioned project ARCHES. In that project we developed DTPs of five ocean observation systems constructed at AWI and GEOMAR. They vary in construction, payload, and configuration (see Barbie et al. [6] for details). The distance between AWI and GEOMAR are a few hundred kilometers. Hence, we required a method to develop the software for all the systems and test the underwater network, without a permanent connection to the physical ocean observation systems. To develop the software system for the PT we primarily utilized open-source tools to build a microservice architecture [5] based on the Robot Operating System (ROS) encapsulated in Docker containers and use this as the basis to run a virtual instance as DT. The underwater communication between PT and DT was established via acoustic modems. The PTs sent sensor data and status updates to the corresponding DTs running on a server on the research vessel. Commands to operate the PTs were sent from the DTs to the PTs.

A disadvantage of SiL testing is the missing ground truth that the data during the tests can really be used to verify and validate the behavior of an embedded software system. In our domain, we are unable to even observe the actions of the PT underwater. Often SiL tools provide software interfaces that receive a data stream and execute simulations on a (mathematical) model or on mock-ups. Interfaces, such as serial connections, are not considered in these tests and small deviations in the underlying model may falsify the tests. Furthermore, the used data is often synthetic, since it solely was created for a specific test case to verify a specific behavior. In ARCHES, our emulators use existing recordings/data from previous missions as ground truth to increase the confidence in the results. However, the recordings of sensors and actuators used by DTPs in general may originate from observations and simulations.

We evaluated this approach during the research cruise AL547 with RV ALKOR (October 20-31, 2020), where we established and deployed a collaborative underwater network of ocean observation systems in the Baltic Sea. During that

period, different scenarios were executed to demonstrate the feasibility of digital twins for maritime environments [6].

With DTPs we were able to develop and test these scenarios before the mission took place. We use GitLab as CI/CD platform. For integration tests, we utilize the testing tools provided by ROS. During the mission, we recorded all exchanged message on the PT and DT and are now able to use this data in our integration tests to increase the quality of our CI/CD pipelines.

Conclusion

We presented the Digital Twin Prototype as fourth subcategory of digital twins. A DTP can be employed by software engineers for embedded software development without a connection to a physical test rig to enable automated SiL testing in CI/CD pipelines. In the development phase of embedded software systems, DTPs pave the way for a paradigm switch from traditional embedded software development processes, e.g. the V-Model, to agile and incremental workflows with CI/CD. This way, DTPs allow a rapid prototyping of embedded software systems and the integration of feedback by stakeholders at a very early development stage.

We designed this method and a software framework, based on ROS, to run digital twins in the context of the project ARCHES and demonstrated the feasibility for ocean observation systems in a real life mission in the Baltic Sea in October 2020 [6].

By using DTPs for the software development, we were not only able to bridge the geographical distance between the AWI and GEOMAR during the COVID-19 pandemic, we were also able to work from our home offices and hence, reduced physical contacts in the team.

All in all our approach increases the quality of embedded software systems and helps to reduce costs while increasing development speed, which has been reported by both Ebert [3] and Ozkaya [2] as reasons for the struggle to achieve quality along with managing costs and efficiency.

Acknowledgment

The project is supported through the HGF-Alliance ARCHES – Autonomous Robotic Networks to Help Modern Societies and the Helmholtz Association.

References

[1] Phillipe Goupil. AIRBUS State of the Art and Practices on FDI and FTC. *IFAC Proceedings Volumes*, 42(8):564–572, 2009. 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes. doi:10.3182/20090630-4-ES-2003.00094.

[2] Christof Ebert. 50 Years of Software Engineering: Progress and Perils. *IEEE Software*, 35(5):94–101, 2018. doi:10.1109/MS.2018.3571228.

[3] Ipek Ozkaya. Can We Really Achieve Software Quality? *IEEE Software*, 38(03):3–6, may 2021. doi:10.1109/MS.2021.3060552.

[4] Vahid Garousi, Michael Felderer, Çağrı Murat Karapıçak, and Uğur Yılmaz. What we know about testing embedded software. *IEEE Software*, 35(4):62–69, 2018. doi:10.1109/MS.2018.2801541.

[5] Alexander Barbie, Wilhelm Hasselbring, Niklas Pech, Stefan Sommer, Sascha Flögel, and Frank Wenzhöfer. Prototyping Autonomous Robotic Networks on Different Layers of RAMI 4.0 with Digital Twins. In *Proceedings of the 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2020)*, pages 1–6. IEEE, 2020. doi:10.1109/MFI49285.2020.9235210.

[6] Alexander Barbie, Niklas Pech, Wilhelm Hasselbring, Sascha Flögel, Frank Wenzhöfer, Michael Walter, Elena Shchekinova, Marc Busse, Matthias Turk, Michael Hofbauer, and Stefan Sommer. Developing an Underwater Network of Ocean Observation Systems with Digital Twin Prototypes – A Field Report from the Baltic Sea. *IEEE Internet Computing*, 2021. doi:10.1109/MIC.2021.3065245.

[7] Roberto Saracco. Digital Twins: Bridging Physical Space and Cyberspace. *Computer*, 52(12):58–64, 2019. doi:10.1109/MC.2019.2942803.

[8] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018. doi:10.1016/j.ifacol.2018.08.474.

[9] Sanket Bachuwar, Ardashir Bulsara, Huzefa Dossaji, Aditya Gopinath, Chris Paredis, Srikanth Pilla, and Yunyi Jia. Integration of Autonomous Vehicle Frameworks for Software-in-the-Loop Testing. *SAE International Journal of Advances and Current Practices in Mobility*, 2:2617–2622, 2020. doi:10.4271/2020-01-0709.

[10] Elisa Yumi Nakagawa, Pablo Oliveira Antonino, Frank Schnicke, Thomas Kuhn, and Peter Liggesmeyer. Continuous Systems and Software Engineering for Industry 4.0: A disruptive view. *Information and Software Technology*, 135:106562, 2021. doi:10.1016/j.infsof.2021.106562.

[11] Wilhelm Hasselbring, Sören Henning, Björn Latte, Armin Möbius, Thomas Richter, Stefan Schalk, and Maik Wojcieszak. Industrial DevOps. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 123–126, March 2019. doi:10.1109/ICSA-C.2019.00029.

Alexander Barbie is a software engineer at the GEOMAR Helmholtz Centre for Ocean Research Kiel (Germany) and the Alfred-Wegener-Institute Helmholtz Centre for Polar and Marine Research (Bremerhaven, Germany), and a doctoral researcher in the software engineering group in the Department of Computer Science, Kiel University, Germany. Contact him at abarbie@geomar.de.

Wilhelm Hasselbring is a full professor of software engineering in the Department of Computer Science at Kiel University, Germany. Contact him at hasselbring@email.uni-kiel.de.

Niklas Pech is a software engineer at the GEOMAR Helmholtz Centre for Ocean Research Kiel (Germany). Contact him at npech@geomar.de.