Towards Modular Digital Twins of Robot Systems

Daniella Tola, Till Böttjer, Peter Gorm Larsen, and Lukas Esterle
Department of Electrical and Computer Engineering and *DIGIT*
Aarhus University
Denmark
{dt,till.boettjer,pgl,lukas.esterle}@ece.au.dk

***Abstract*—Robots are used in various types of applications in manufacturing, ranging from assembly to machine tending. Changes in production lines are becoming more common, either due to product changes, requiring changing manufacturing processes, as well as faults, caused by wear and tear of the robots. In such cases it is crucial that the robot system can adapt to the new circumstances. Recently the concept of Digital Twins (DTs) has gained a large amount of interest in both academia and industry. The idea of a DT is to have a digital copy of a physical system, where both systems exchange information which can be used to adapt the system to unforeseen changes in itself or its environment. This enables a DT to perform self-adaptation when problems arise during operation. However, development of a DT, whether for existing or future robots, is cumbersome. We therefore propose a development of modular DTs and outline our methodology for creating DTs for robot systems, their shortcomings and open challenges. Such a modular DT can be utilised to safely explore the potential adaptations of the physical system and its performance in changing environmental conditions.**

***Index Terms*—digital shadow, robotic arms, modularity, visualisation**

I. Introduction

Industrial robots, and robot systems in general, become more modularised, often combining a potentially mobile base platform, end effector(s), data connections, and robotic arm(s) of various lengths [1] (see Fig. 1). This modularity offers great flexibility for robot system designers and even allow the robot to self-adapt to changing requirements by replacing individual components (e.g. changing the end effector). The end effector is connected via an end effector coupling device (EECD) to the robotic arm and used for executing the task. This is necessary as the individual components in a robot system are often produced by different manufacturers, with the exception of the EECD and end effector, which are typically produced by the same manufacturer.

Over the past years, Digital Twins (DTs) [2] have gained popularity. DTs provide accurate models of physical systems. In contrast to computer models, DTs can exchange information bi-directionally during runtime. This enables the users to observe the behaviour of the system but also to control the physical system through the DT. Digital Shadows (DSs) present a necessary preliminary stage of DTs, while only implementing the one-way, physical-to-virtual connection. Thus, changes of the physical entity are mapped to the virtual entity but not vice versa [2].

Furthermore, DTs have traditionally been utilised for predictive maintenance, fault and error prediction, runtime verification, and various safety operations for individual components and entire robot systems. However, designing DTs is usually done for a single, consistent robot system. With rising modularity of robot systems and robots operating in dynamic environments expected to adapt themselves to changing conditions, more flexible DTs are required as well. Specifically in SISSY settings, this flexibility of DTs has several benefits:

- Self-improvement through exploration of potential changes to the physical robot and its impact on the performance before executing the adaptation.
- Identification of changes in the environment or on the robot due to a performance drift between physical system and DT.
- Verified adaptation by combining pre-verified DT modules.

In this paper we illustrate the first step towards modular DTs. The main contributions of this paper can be summarised as:

- (Section III) We propose an architecture to aid development of flexible and modular DTs.
- (Section IV) We illustrate the approach through a case study where we develop two DSs of different robot systems.
- (Section V) We outline how we believe to move forward in the development of modular DTs of robot systems by describing main technologies in this field.

Section II provides an overview of DT concepts in robot systems and discusses related work and the main advantages of using DTs for robot systems. Concluding remarks are presented in Section VI.

II. Overview of Digital Twins for Robot Systems

A DT consists of a physical entity, a digital entity and a bi-directional communication. The physical entity sends data to the virtual entity, which analyses the data and returns information to the physical entity. This section describes the related work in the area of digital twins of robot systems, and the main advantages presented by others for creating DTs of robot systems.

We would like to thank the Innovation Foundation Denmark for the MADE FAST project and the Aarhus University Digital Transformation Lab. We would also like to thank Santiago Gil Arboleda for helping with the development of the communication with the Kuka robot, and the anonymous reviewers for the detailed feedback.

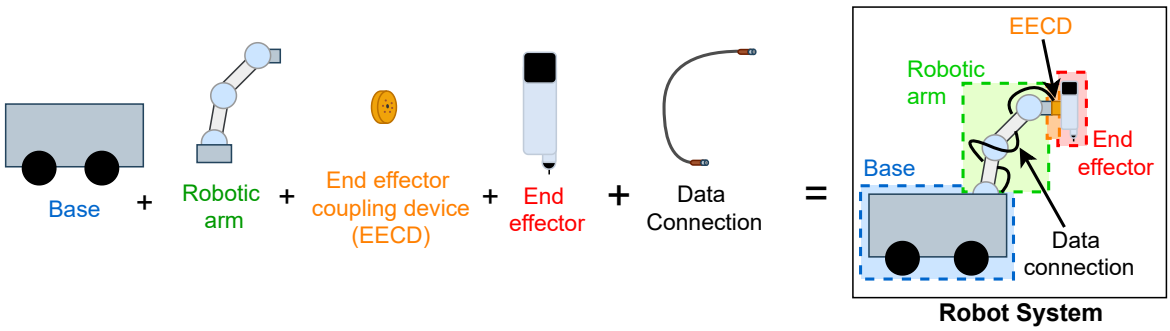


Fig. 1: Most common devices in a robot system.

A. Related Work and Technologies

Hoebert et. al. [3] present a DT framework for industrial robot systems focused on assembly planning. The key feature of the framework is a semantic ontology description created in Neo4j. It describes the robot and workpiece geometry, and assembly actions linked to required tools for individual operations. The virtual entity is implemented in HTML-5. The communication is implemented via HTTP and JSON. 3D real-time rendering is implemented using BabylonJS tool and WebGL. The framework is validated on a KUKA robot assembling various components on a PCB board.

Arkouli et. al. [4] present a lumped parameter model considering joint flexibility of a robot manipulator. They describe a method for calibrating the model parameter through an iterative calculation of the error between measurement and simulation data.

Barenji et. al. [5] present a DS for a robot manipulator, they: (i) create a 3D CAD model, (ii) generate a URDF file, (iii) derive kinematics, and (iv) plan trajectories with respect to minimum energy while avoiding collisions.

Huynh et. al. [6] present a DS to simulate robot movement in a web application. They created this using the Python Flask micro framework and the JavaScript API, WebGL. A model of the manipulator was created using a json file to describe the kinematics of the robot. The presented application has a considerable delay of over 1 second when visualising the movement of the physical robot.

These studies present different frameworks, technologies and tools towards creating DTs of robot systems. The majority of the research reports on creating models for robot systems and uses DSs for the visualisation of the movement of physical robots in the virtual space. However, they do not describe how to modularise the approach for robot systems.

B. Advantages of Digital Twins of Robot Systems

While DTs can be developed for robotic cells and lines with several robots and auxiliary equipment, we constrain this research to a single robot system, focusing on services related to a single robot and not on supply chain and production planning related aspects. The main advantages of DTs for robot systems are optimisation and improvement of the automated processes themselves, while DTs of production lines focus more on the optimisation of planning and scheduling between multiple lines.

DTs of collaborative robot systems are often used for visualisation and virtual commissioning. Havard et. al. [7] use a DT to improve the layout and ergonomics of a workstation, while ensuring the safety of the human operator. Baskaran et. al. [8] show how a DT can help to find the optimum assembly sequence. Malik et. al. [9] discuss the benefits of reducing development costs and improving safety through DTs. Fernández et. al. [10] test and validate configurations of a robot system through simulation of real-world scenarios using historic data of existing DTs.

Further, DTs of robots systems can help avoiding collisions by adapting their trajectories. Tammato et. al. [11] and Huynh et. al. [6] use DTs to visualise objects entering and leaving the working area of a robot and to plan timely intervention and re-calculation of motion trajectories.

Also, robotic DTs can support planning of maintenance activities related to component failure. Aivaliotis et. al. [12] predict the remaining useful lifetime of robot joints. Park et. al. [13] classify failures of a robot system through a multi-model DT approach.

III. Modularising Digital Twins for Robot Systems

Robot systems are themselves modular, to a certain extent, as it is possible to plug and play a number of products from different manufacturers. As Fig. 1 illustrates, a robot system is composed of a number of devices produced by various manufacturers. The devices can be configured into a robot system if they have compatible interfaces.

An approach towards modular DTs is established by identifying the main components within DTs of robot systems; defining their interfaces and interconnections, and describing domain knowledge.

The proposed architecture is illustrated in Fig. 2, where the main modules and their connections are presented. There can be multiple services in such a DT and depending on the services, different types of models may be used and in some cases none at all.

We define the main modules in a DT of a robot system to be, the following (their connections can be seen in Fig. 2):

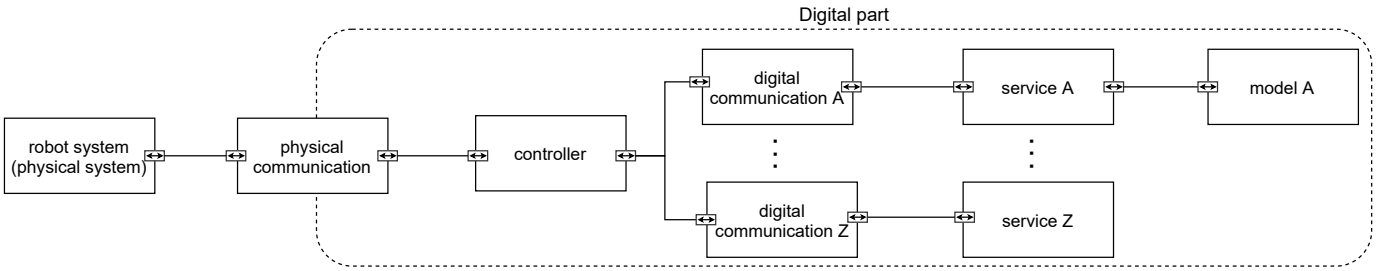


Fig. 2: Proposed architecture for modular digital twins of robot systems.

- **robot system:** the physical system where the main devices are the robotic arm and end effector. For these devices to connect to the communication module of the DT, they must have a compatible interface.
- **physical communication:** ensures a connection between the robot system and the digital part to allow flow of self-adaptation information and data.
- **controller:** responsible for ensuring relevant data exchange between the robot system and the services. This is the main controller in the DT where all the data that is exchanged passes through and is processed.
- **digital communication:** used to communicate information between the controller and the services. Each service may support a different communication protocol, which the compatibility with the controller depends on.
- **services:** adds value to the DT by providing visualisations, predictive maintenance, optimisation, root cause analysis, etc. As illustrated in Fig. 2 some services require models.
- **models:** typically represent a part of the system and are used in services. Different types of models can be used, e.g. kinematic, dynamic, etc.

IV. Case Study

We illustrate how to use the architecture by creating prototypes of two DSs. To ensure modularity across manufacturers, we chose to use robotic arms from two different manufacturers and two types of end effectors. The physical and digital robots are shown in Fig. 3.

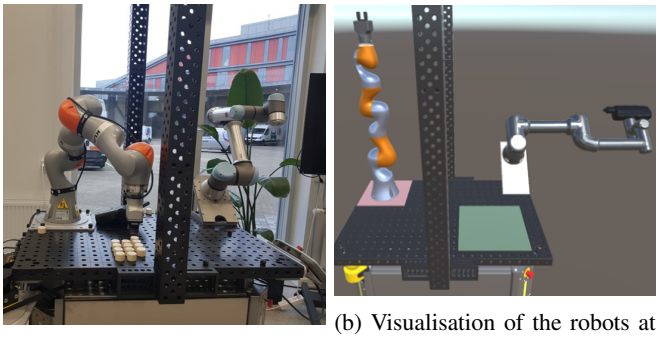
An overview of the implementation is illustrated in Fig. 4, where each of the parts in the implementation are described in the following sections. The main properties of the DSs are presented in Table I. The time lag presents how long it takes on average to send data from the controller and receive it on the visualisation service. The inputs to the service are currently only the joint angles of the robotic arms, used for visualisation purposes. However, any sensor information or data available on the robot can be sent to the services as inputs.

A. Robot System (Physical System)

We refer to the case studies as *Robot System A* and *Robot System B*. *Robot System A* is the *Kuka Lbr Iiwa R800* manufactured by Kuka, and has seven degrees of freedom. It has a two-finger gripper, specifically the 2FG7 Parallel Gripper from Onrobot[1], attached to it as its end effector for demonstrating a Pick and Place application. *Robot System B* is the *UR5e* manufactured by Universal Robots and has six degrees of freedom. It has the Onrobot Screwdriver[2] attached to it as its end effector for demonstrating a screwdriving application. The robot systems can be seen in Fig. 3, with Kuka 3.a (left) and UR5e 3.a (right).

Properties	Values
Time lag	Less than 500 ms on average
Sensors	Built-in robotic arm
Inputs to service	Joint angles of robotic arms
Models	Kinematic (URDF)
Software	Unity, ZeroMQ, Python

TABLE I: Overview of the main properties of the digital shadows.



(a) Picture of the robots at DTL.

(b) Visualisation of the robots at DTL.

Fig. 3: Illustration of the visualisation service of the two robot systems.

B. Physical Communication

A standardised communication interface for robot systems is lacking, as stated by Sannemann et al. [14], making it difficult to modularise the communication between a physical and a digital counterpart. Creating physical communication modules that are modular is a complicated task when working with robot systems, as they typically need to be physically connected and configured using an IP address.

The amount of work carried out in the development of libraries for communicating with robotic arms varies sub-

[1] onrobot.com/en/products/2fg7

[2] onrobot.com/en/products/onrobot-screwdriver

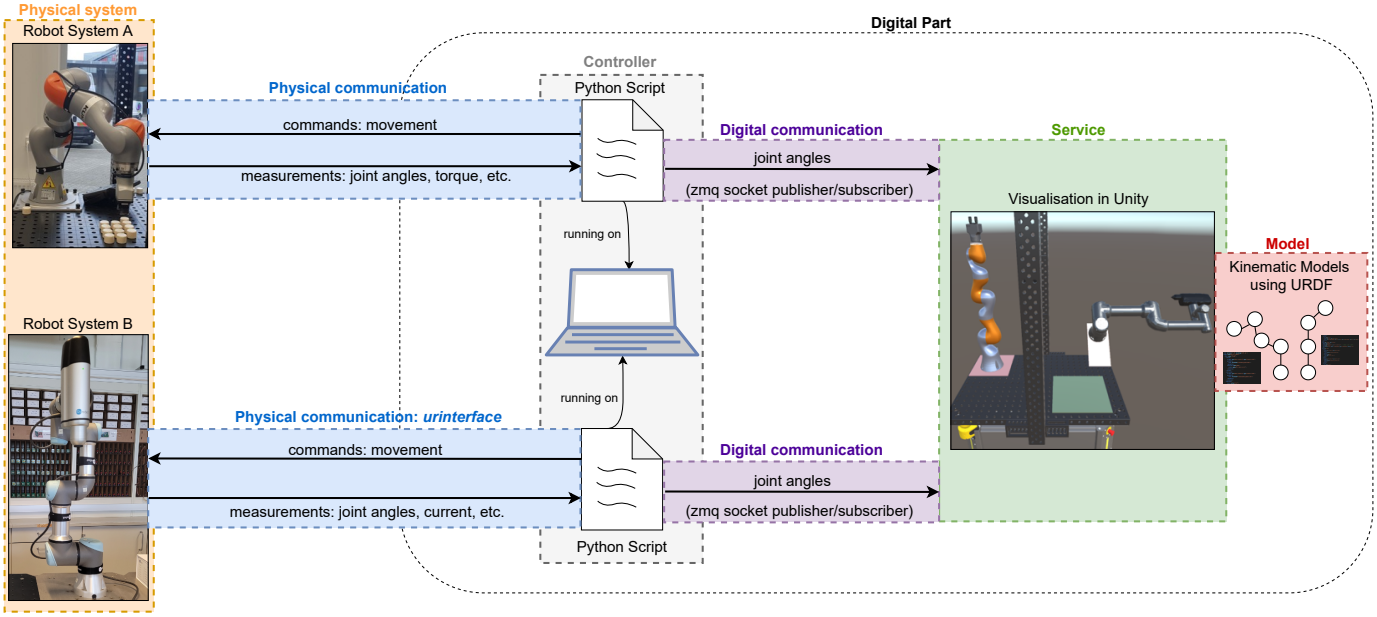


Fig. 4: Overview of the implementation of the DSs of the case studies based on the proposed architecture in Fig. 2.

stantially depending on the manufacturer. In many cases, researchers or the industry have created third party software to communicate with robotic arms. As an example, *Universal Robots* has created the *Real-Time Data Exchange (RTDE)*[3] protocol for interfacing with their robots. Multiple libraries have been developed, building on top of RTDE, such as the `urinterface`[4] or the `ur_rtde`[5]. These libraries are open-source, allowing others to utilise them when communicating with robotic arms from *Universal Robots*. Other robotic arms, such as *Kuka* robots, require more work to communicate with. Researchers have developed a Matlab toolbox for communicating with a *Kuka* robotic arm [15]. Other researchers have created a cross-platform communication interface for *Kuka* robots [16]. Apart from libraries developed for specific robots, the *Robot Operating System (ROS)* middleware contains a *ROS Industrial*[6] package with a number of libraries for robotic arms. Unfortunately, most of these *ROS Industrial* libraries are still in experimental stages.

Our Implementation. The physical communication to the two robots is implemented using two different frameworks. The communication with the Kuka Lbr Iiwa was developed in our lab, inspired by the work carried out in [17]. A Java client was programmed on the Kuka controller and a Python server was set up on the DT controller. The communication between the controller and the UR5e was implemented using the python framework `urinterface`. The data recorded from both robots were the joint angles. Both robotic arms were controlled by sending movement commands using the communication frameworks. Note, that these commands were implemented as predefined movements that the robot executed at run-time. This means there was no feedback-loop in the control of the physical robot.

The physical communication can be accomplished using different communication protocols depending on the type of robot. Creating a framework similar to `urinterface` for robots from different manufacturers and of different types is beneficial, especially if such frameworks are combined into a single tool. The tool would allow to easily programm different types of robots from the same interface and is a great method for modularising the physical communication. The physical connection would be through an ethernet cable which typical robotic arms support. A similar method used by the program *drag&bot*[7] is to connect a computer to the digital I/O ports of a robot controller for configuring and programming the robot. However, this is a more tedious process as each robot has different requirements for the configuration and programming.

C. Digital Communication

We use an approach of publisher and subscriber modules, for communicating between the controller and services. The publisher module sends data to a topic that any module can subscribe to for constant information exchange. We set up a `zeromq`[8] socket between the controller and the Unity[9] visualisation, where the controller publishes the data received from the physical system to a topic that the service subscribes to.

[3] universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide
[4] pypi.org/project/urinterface
[5] sdurobotics.gitlab.io/ur_rtde/index.html
[6] http://wiki.ros.org/Industrial
[7] www.dragandbot.com
[8] https://zeromq.org/
[9] unity.com

D. Controller

Two controllers have been developed, one for each DS. Each controller is implemented as a Python script that:

- sends predefined movement commands to the robot,
- receives measurement data from the robot,
- and publishes the joint angles to a port through a `zeromq` socket

The controllers were both running on the same computer, allowing us to combine the data into one visualisation.

E. Models and Services

Most services that can be used in a DT require some type of model. For example, a service for energy optimisation or fault detection would most likely require a model of the currents powering the motors at each joint [5]. Another example is presented by Bansal et al. [18] where they use kinematic models in a simulation to detect objects and avoid collisions.

In this paper we focus on the visualisation service of the DT, and illustrate it using kinematic models of the robots and the game-engine Unity. One of the first steps towards creating a DT is to set up the services for the system. In this case, we set up a 3D visualisation of the robot systems, which currently is a mirrored visualisation of the physical system.

The kinematic models of the UR5e and Kuka Lbr Iiwa were developed by creating Unified Robot Description Format (URDF)[10] files for each robot. The URDF is an XML formatted file representing a robot model [19], by defining the position of the links (rigid bodies) and the joints that connect the links. The URDF file allows defining meshes for 3D visualisation, and the mass and inertia for simulation purposes. URDF files are standalone and can be imported in many different programs, such as Unity, RViz, etc., making them easy to reuse in other services.

URDF files for a limited number of robots can be found online on GitHub repositories such as `ros-industrial`[11]. Unfortunately, it is not yet common practice for manufacturers of robot devices to make such files publicly available, meaning in many cases developers themselves must create these files. In such cases the accuracy of the models may be affected. Creating a public database or repository where such models can be uploaded may accelerate the process of creating modular DTs for robot systems, as it allows developers to easily access and utilise models instead of wasting time on developing them themselves. To initiate this idea, we have created a public GitHub repository and added our current models of the robot systems for others to use. The repository can be found in the link[12].

The visualisation in Unity of this case study is shown in Fig. 3b. A recording of the visualisation of a different system can be found in the link[13] illustrating how the DS works. Using Unity allows us to extend the visualisations to Augmented and Virtual Reality with various advantages. An example of a future use could be to visualise important process parameters or values such as the heat dissipated in a motor, allowing technicians to rapidly localise faults in a system.

[10]http://wiki.ros.org/urdf

[11]https://github.com/ros-industrial/universal_robot

[12]https://github.com/Daniella1/robot_urdfs

[13]https://youtu.be/1vt7-qrFvZc

V. Future Developments Towards Digital Twins in SISSY Systems

SISSY systems operate in open environments with dynamic changes and are expected to self-improve their integration with the environment and others [20]. Considering SISSY systems to be cyber-physical systems, DTs can support the system to perform exploration of potential adaptions without directly implementing them in the physical system. Upon detecting such an improvement in the digital system, the physical system can also implement and utilise this improvement. However, with traditional DTs, the system can only integrate and improve as a whole [21]. Here, the modularity of robotic systems gives rise to the benefits of modular DTs as they can support exploration of changes within the individual system. Now a robot can safely explore physical changes that may lead to improved performance given a changed environment at runtime.

In our current implementation, the approach is not yet fully modular, and therefore future work would be to develop a framework or tool to ease the setup of modular DTs. As our current implementation is a DS, it needs to be extended to a DT by adding self-adaptation information flow from the digital to the physical system. Communication from the digital to the physical system has already been established, the main part that is missing is the automatic update of the physical system enabling self-adaptation. The services must also be expanded, where we plan to use `aurt` for calibrating the dynamics of the robotic arms [22]. This self-improvement mechanism can potentially be used for anomaly and collision detection in a self-adaptation loop. Assuming the extension is to perform online collision detection and avoidance, then the information from the controller to the physical system would be an updated collision-free trajectory.

One of the approaches we are working towards is using co-simulation for detecting anomalies and sending information to the physical system on how to fix the issues. An example where co-simulation has been used in a DT with a self-adaptation loop is presented by Feng et al. [23]. Co-simulation is the process of simulating models developed using different tools. One of the most common standards used in the field of co-simulation in the Functional Mock-Up Interface (FMI) [24] which allows encapsulation of models as black boxes. Different Original Equipment Manufacturers (OEMs) can provide black box models of their devices using the FMI standard, and at the same time protect their Intellectual Property. Such models that implement the FMI standard are called Functional Mock-up Units (FMUs). These black box models can then be used in a co-simulation, where virtual scenarios of robot systems can be assembled. Co-simulation allows simulating these FMI-based black box models which can be developed using different tools [25].

A direction to work towards together with the industry would be to create a public database with FMUs containing black box models of robots, their controllers and so on. If the manufacturers of such robot devices uploaded FMUs to a shared repository, then developing DTs of their products would be a less demanding process. Currently, developers themselves need to develop controllers for the robots if they want to simulate them, which means the simulation of the robot and its controller is an approximation of how the physical robot would actually move. To achieve a simulation as close to reality as possible, it is beneficial to use high fidelity models developed by the manufacturers of the devices. We have already developed a public method for creating FMUs using Unity, and thus are already a step towards creating modular DTs of robot systems and using visualisations as a service[14].

VI. Concluding Remarks

In this paper we proposed an architecture for developing modular digital twins that can be used in the development of self-adaptive systems. We validated the approach by implementing digital shadows of two different robot systems for displaying the modularity of the approach. The implementation of the digital shadows is uni-directional from the physical system to the digital system. However, we plan to extend the implementation to a digital twin where a self-adaptation loop will be developed. This will be based on calibrating the dynamics of the robotic arm, using `aurt`, and optimising the movement of the robot when the motors start to show signs of wear.

References

[1] "Robotics — vocabulary," International Organization for Standardization, Geneva, CH, Standard, 2021.

[2] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[3] T. Hoebert, W. Lepuschitz, E. List, and M. Merdan, "Cloud-Based Digital Twin for Industrial Robotics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11710 LNAI, pp. 105–116, 2019.

[4] Z. Arkouli, P. Aivaliotis, and S. Makris, "Towards accurate robot modelling of flexible robotic manipulators," in *Procedia CIRP*, vol. 97. Elsevier B.V., 2020, pp. 497–501.

[5] A. Vatankhah Barenji, X. Liu, H. Guo, and Z. Li, "A digital twin-driven approach towards smart manufacturing: reduced energy consumption for a robotic cellular," *International Journal of Computer Integrated Manufacturing*, 2020.

[6] B. H. Huynh, H. Akhtar, and M. K. Sett, "A universal methodology to create digital twins for serial and parallel manipulators," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 3104–3109.

[7] V. Havard, B. Jeanne, M. Lacomblez, and D. Baudry, "Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations," *Production and Manufacturing Research*, vol. 7, no. 1, pp. 472–489, 2019.

[8] S. Baskaran, F. A. Niaki, M. Tomaszewski, J. S. Gill, Y. Chen, Y. Jia, L. Mears, and V. Krovi, "Digital human and robot simulation in automotive assembly using siemens process simulate: A feasibility study," in *Procedia Manufacturing*, vol. 34. Elsevier B.V., 2019, pp. 986–994.

[14] github.com/INTO-CPS-Association/unifmu_examples/tree/master/examples/UnityFMUTemplate

[9] A. A. Malik and A. Brem, "Digital twins for collaborative robots: A case study in human-robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 68, 2021.

[10] I. A. Fernández, A. Aguirre Ortuzar, J. U. Andrés, and L. Eciolaza Echeverría, "ADAPT: An Automatic Diagnosis of Activity and Processes in auTomation environments," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2020-Septe. Institute of Electrical and Electronics Engineers Inc., 2020, pp. 945–951.

[11] A. Tammaro, A. Segura, A. Moreno, and J. R. Sánchez, "Extending industrial digital twins with optical object tracking," *Eurographics Symposium on Geometry Processing*, vol. 36, no. 2, pp. 23–26, 2017.

[12] P. Aivaliotis, K. Georgoulias, and G. Chryssolouris, "The use of Digital Twin for predictive maintenance in manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 11, pp. 1067–1080, 2019.

[13] H. Park, A. Easwaran, and S. Andalam, "TiLA: Twin-in-the-loop architecture for cyber-physical production systems," in *Proceedings - 2019 IEEE International Conference on Computer Design, ICCD 2019*. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 82–90.

[14] L. Sanneman, C. Fourie, and J. A. Shah, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *CoRR*, vol. abs/2010.14537, 2020.

[15] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "Kct: a matlab toolbox for motion control of kuka robot manipulators," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 4603–4608.

[16] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen, "Controlling kuka industrial robots: Flexible communication interface jopenshowvar," *IEEE Robotics & Automation Magazine*, vol. 22, pp. 96–109, 2015.

[17] M. Safeea and P. Neto, "Kuka sunrise toolbox: Interfacing collaborative robots with matlab," *IEEE Robotics Automation Magazine*, vol. 26, no. 1, pp. 91–96, March 2019.

[18] R. Bansal, M. A. Khanesar, and D. Branson, "Ant colony optimization algorithm for industrial robot programming in a digital twin," in *2019 25th International Conference on Automation and Computing (ICAC)*, 2019, pp. 1–5.

[19] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*, 1st ed. O'Reilly Media, Inc., 2015.

[20] K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. R. Lewis, P. R. Nelson, E. Pournaras, A. Stein *et al.*, "Self-improving system integration: Mastering continuous change," *Future Generation Computer Systems*, vol. 117, pp. 29–46, 2021.

[21] L. Esterle, C. Gomes, M. Frasheri, H. Ejersbo, S. Tomforde, and P. G. Larsen, "Digital twins for collaboration and self-integration," in *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE, pp. 172–177.

[22] E. Madsen, D. Tola, C. Hansen, C. Gomes, and P. G. Larsen, "Aurt: A tool for dynamics calibration of robot manipulators," in *2022 IEEE/SICE International Symposium on System Integration (SII)*, 2022, pp. 190–195.

[23] H. Feng, C. Gomes, S. Gil, P. H. Mikkelsen, D. Tola, M. Sandberg, and P. Larsen, "Integration of the mape-k loop in digital twins," in *Annual Modeling and Simulation Conference (ANNSIM 2022)*, 2022.

[24] Modelica Association, "Functional Mock-up Interface for Model Exchange and Co-Simulation," https://www.fmi-standard.org/downloads, October 2019.

[25] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey," *ACM Comput. Surv.*, vol. 51, no. 3, May 2018.