Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 51 (2020) 1799–1805

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

A Digital Twin Creation Method for an Opensource Low-cost Changeable Learning Factory

Tarek Al-Geddawy*

*Manufacturing Engineering, Western Wshigton University, Bellingham, WA, USA

* Corresponding author.

Abstract

Learning factories demonstrate applications and technology to students in a real industrial environment. While turnkey changeable learning factories are supplied by many vendors, with some digital twin capabilities, they are mostly a closed box, with very little flexibility to change the underlying architecture or technology, hindering the maximum benefit of student hands-on experience. This paper presents a method to build a simulated changeable learning factory and link it to the physical system to create a digital twin. The studied learning factory (LEAF) is an opensource low-cost changeable automated system. The suggested digital environment is 'RoboDK', which is a 3D simulation and offline/online programming environment, mainly for industrial robots, but it also offers an open source 'Python' programming library, allowing the extension of the capabilities of the software to adapt to LEAF. The method is also using the opensource Modbus TCP and OPC UA industrial communication protocols to establish the connection between the physical modules and the digital objects. The results show a capable digital system that is accurately mirroring the physical system layout and material flow, with a flexible structure to allow future extensions.

Keywords: Learning Factory; Digital Twin; Changeable Manufacturing

1. Motivation

Learning factories are used to demonstrate industrial applications to engineering students, trainees and researchers in a real industrial environment. This introduces current industrial challenges to research labs, and mimics larger scale and more complex manufacturing and logistics systems. Those experimental, teaching and research facilities provide an excellent environment to develop, test and implement new product designs and system concepts [1]. They can be described as a factory-in-a-lab [2]. Those facilities are used to educate STEM students and researchers about product emergence processes, logistics optimization, management and organization, business administration and automation technology [3]. However, five main limitations of learning factories have been identified [4]; needed resources, mapping ability of different industrial issues, class scalability, mobility, and effectiveness. Learning factory design has been identified as the main factor that impacts those limitations, specially mapping, mobility and effectiveness.

The design of the state-of-the-art learning factories is changeable. A Changeable Learning Factory (CLF) is flexible, reconfigurable, modular and sustainable to enable researchers to study a wide range of case studies and production scenarios within an enterprise simulated environment [5]. Technically, changeability is the quick and efficient response to change in demands, design, and technology. Its main goal is to economically accomplish early and foreseeable adjustments of the enterprise structure and processes on all levels [6]. The configuration layout of a CLF should follow a modular approach to allow flexibility on its application and operation. Multiple layout options should be possible when different

10.1016/j.promfg.2020.10.250

modules are being combined. Such configurations could accommodate multiple knowledge receivers [7].

The focus of this paper is using learning factories in the field of automation. The education process of industrial automation and integrated systems has multiple levels of granularity, parallel to the granularity level of the physical automated systems. The lowest granularity level starts at working with individual components, such as sensors, actuators, robots, vision systems, etc. then controlling them with PLCs and drives, and building a network of controllers and PLCs in a distributed control system (DCS). Moving up in the granularity level would be teaching students about industrial communication protocols, integrating Human-Machine-Interface (HMI) and building a Supervisory Control and Data Acquisition (SCADA) system. On top of that knowledge comes the implementation of Industry 4.0 and IIoT. The hands-on experience in such learning process is crucial, specially working with industrial grade components, and with a neutral approach toward technology, i.e. not focusing on a specific automation supplier. Using a learning factory should satisfy most of these educational granularity levels, especially if it is a changeable system that can be split in smaller separate units for smaller groups of students to cover lower level hands-on experience and knowledge.

Turnkey learning factories are offered from many vendors, especially in the field of automation technology, such as the iFactory in University of Windsor and University of Stuttgart, which is developed by Festo Didactec [8]. They are mostly scalable to accommodate available financial resources with some room to grow and expand, following industry standards specially in safety, and some of them are modular with reconfigurable layout. Since learning factories have also the goal to develop competency in students and trainees [4], an understanding of the underlying technology and system architecture should be promoted. The best way to do that is by allowing the physical hands-on experience of designing, assembling and disassembling such systems. This option is not available for most existing turnkey learning factories, since vendors would not allow altering the physical system underlying technology, risking system functionality, safety and warranty. On the other hand, allowing the full hands-on experience of developing a large integrated system is a complex process, especially if it comes with many stations and processes of different manufacturers, some of which come as a surplus from industrial donors to academia. Using a turnkey highly modular CLF may look simpler on the operational level, but it is very complex on the structural level, through which students develop their deep understanding of the system.

This paper presents an open-access approach to create a digital twin to an existing open-access physical system. Both systems have been developed and built by students to help them learn the subjects of automation and system integration, while keeping cost and investment as low as possible. This approach can help encourage other institutes to develop their own systems. The paper is divided into the following sections; a literature survey of the some of the existing digital environments and a comparison of their cost and openness, a presentation of the used digital components in the presented method, a presentation of integration method with an existing learning factory, and finally discussion and future work.

2. Literature Survey

2.1. Digital Twin Definition

The continuous digitization of manufacturing processes and automation has paved the road to the implementation of digital simulation and emulation of processes and systems. The digital twin is a presentation of characteristics and behavior of a system according to various levels of details and the scope it addresses [9]. The digital twin is a virtual clone of a whole or a part of a factory [10]. Digital 'simulation' is a forecast of running the process given a set of parameters, while 'emulation' is building a whole soft system and system behavior on a computer. The collection of these simulated and emulated instances constitutes the 'Digital Twin' of a system.

2.2. Case studies from literature

Very few examples of detailed digital twin models of learning factories exist in literature. A digital twin of a single Yaskawa Motoman robot cell was created in Unity3D game engine [10]. The communication between the physical and the simulated systems was handled through the MQTT protocol, however, a special MQTT script was created to make the Unity engine able to use the MQTT protocol. A KUKA robot cell was simulated using the VEROSIM environment [11]. VEROSIM provides a scripting language called SOML++, which was used to monitor the occurring force and torque during manufacturing simulation. It can be seen that those examples were limited in size, scope and specific in application with many non-opensource components.

2.3. 3D digital environments

One of the main components of the digital twin is the digital environment, which has the visualization aspects and the system components database. For a learning factory, ease-of-use, low cost and being open source are prime features that need to be taken into consideration when selecting such an environment. In addition, the existence of community of users, support and accessible examples are highly valuable merits. There will be a particular focus on digital environments that allow the simulation of mechanisms and robots, since modern automation and systems integration curricula implement robotics in both manufacturing processes and material handling.

Each robot manufacturer has a proprietary digital environment that can communicate, simulate and program its line of robots and perhaps allowing adding other 3D objects and sometimes integration with programming platforms, however these environments are mostly closed-source and only available for those particular robots, such as WinCAPSIII for Denso, RC+ for Epson, RoboGuide for Fanuc, RobotStudio for ABB, Kuka.Sim for Kuka robots, etc.

There are also other general 3D simulation platforms that are not tied to a specific brands, which are candidates for the required digital twin [12]. 3DEXPERIENCE is one of the candidates for such environment. 3DEXPERIENCE is a large connected software platform that can connect engineering applications such as CAD, CAM, Robotics, etc. with other activities such as collaboration and business across an entire enterprise [13]. The robotics application is extensive and has a large library of industrial robots that can be simulated and programmed, however, building a complete system is an extensive work, while the platform itself is not open source. It can be integrated with MATLAB and other programming platforms, but there is a lack of community of users and examples.

Another platform is SimumatiK 3D, which is a free 3D environment that can be used to simulate entire manufacturing systems [14], and has an OPC UA server to communicate with the physical twin, however it is not open source, and can only import ABB robots, with no robot programming capabilities, or clear integration path with programming platforms. Those platforms are exclusively Windows-based software.

On the other hand, the Robot Operating System (ROS) is Linux-based. ROS is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. It has an Industrial ROS extension that is aimed at industrial robots rather than general and mobile robotics, and it is usually paired with Gazebo as a 3D environment representation [15]. ROS has been used extensively in robotics, robot-collaboration [16] and machine vision [17] research because of it is open source and the existence of its large community of users, however, it has a steep learning curve and not user friendly when it comes to quick setup and integration with other systems.

CoppeliaSim (previously known as V-REP) is a cross-platform simulation environment, which means it can be installed on Windows, OS and Linux. It can be integrated with many programming platforms, and has a large ready to use object library including some industrial robots [18], however, it does not have robot post processors for code generation, and not user-friendly for quick system setup since controllers need to be programmed by the user a priory.

RoboDK [19] is also a cross-platform simulation environment that is open source and can be even installed on a RaspberryPI (RPI), which helps using low cost controllers in the learning factory. RoboDK is standalone industrial robot simulation and programming platform that can be integrated with Python, MATLAB and visual studio. Python is distributed with the RoboDK installation, while Python codes are also the native language of the platform that can be executed from inside the simulation environment [20]. It has a large library of objects with industrial robots focus and a simple way to build robot models that are not in the library. The platform also has OPC UA server and client to communicate with the physical twin. RoboDK was the selected digital environment to quickly develop a digital twin for the learning factory. It was also heavily implemented in teaching industrial robotics and systems integration classes.

There exist other pieces of software that can be also used as a digital twin, however, the author does not have access to, such as Siemens process simulate robotics. Five of the digital environments that have a lot of users, tutorials and community support have been discussed. They are also compared in Table 1, which indicates that Industrial ROS and RoboDK have the most needed characteristics for an open-source low cost digital twin in a learning environment, with RoboDK edging over ROS for being more flexible in native 3D environment, cross platforms, extensive robot library and ease of use. Some comparison features such as user-friendliness and learning curve are based on observing and discussing with different group of students in addition to the author's own experience with these different packages. The 'industrial robots focus' feature is useful to get students up and running quickly without pre-building a lot of models, but at the same time, a 'building custom components' capability is useful for adding other mechanisms to the system if needed such as cartesian robots and material handling equipment.

Table 1. 3D Simulation environment feature comparison

Features	**3DEXPERIENCE**	**SimumatiK 3D**	**Industrial ROS**	**CoppeliaSIM**	**RoboDK**
Cross platform	Windows	Windows	Linux+RPI	All	All+RPI
Opensource	No	No	Yes	No	Yes
Cost	Very High	Free	Free	Low	Very Low
Multiple programming languages	Possible w difficulty	No	C++, python and MATLAB	Mainly C++	Integrated python, Visual studio, C++, MATLAB libraries
Industrial Robots Focus	Yes, with extensive robot library	Yes	Yes	No	Yes, with extensive robot library
Physical world communication	Possible w difficulty	OPC UA	Possible	No	OPC UA and python libraries
Community support	Not public	No	Large support	Small	Large support
User-friendly	Yes	Yes	No due to the need to establish nodes and other components first	No due to very small library of ready to use industrial robots	Yes due to large ready to use industrial robots
Learning curve	Slow due to learning multiple applications	Quick due to working in a single visual environment	Slow due to learning multiple components and working under Linux	Quick due to working in a single visual environment	Quick due to working in a single visual environment
Building custom components	Difficult	Difficult	Normal process	Normal process	Simple drag and drop

3. Components of the digital twin model

3.1. LEAF physical hardware structure

The developed open-source low cost learning factory (LEAF) in Western Washington University (Fig. 1) is composed of root and branch material handling modules based on the least system complexity [21], those modules are loop conveyors that circulate pallets on conveyor tops, from root (CABLE) to branches (SABLE), and from conveyors to other processing modules. Current available modules are an Automatic Storage and Retrieval System (ASTRO), a robotic handling with vision module (RAVI), a robotic pick and place and rotating storage module (RANDI) and a robotic assembly module with a SCARA robot (SCAR), in addition to the non-fabrication SCADA unit. The system is reconfigurable, i.e. all modules can be removed, replaced, added and shuffled.

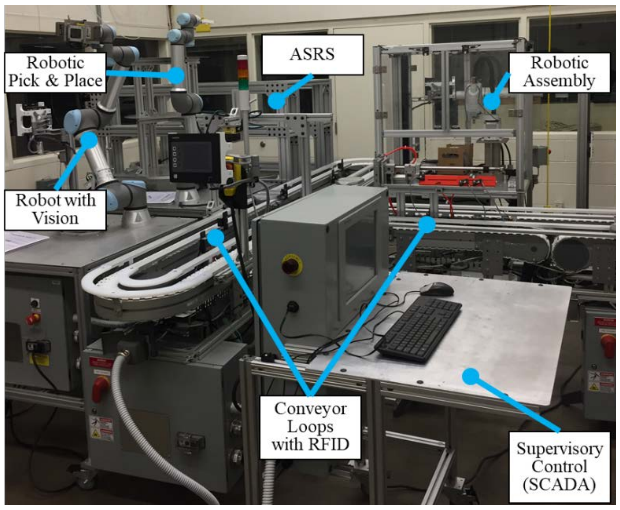

Fig. 1. LEAF physical components at WWU

3.2. LEAF digital environment components

There are four software components that are used in building the digital twin for LEAF in addition to the use of RoboDK, which are CODESYS for control, Modbus and OPC UA for communication, Python for programming.

3.2.1. CODESYS Development and Runtime

The CODESYS [22] Development System is an open-source IEC 61131-3 standard free programming tool for industrial control and automation technology. All 5 control languages can be used in the same code, which are instruction list, structured text, ladder diagram, function block diagram and sequential function chart. In addition, the CODESYS runtime control is a sofPLC that can run on a computer (windows, Linux or a RPI) and works as a standard automation controller with an extended library of industrial communication protocols. The use of CODESYS runtime control streamlines communication between the physical twin and RoboDK. It can be also used as a SCADA system with HMI screens. The runtime control also has a real time version for critical applications.

3.2.2. Modbus Protocol

The Modbus protocol [23] is an open-source industrial communication protocol with a simple and well-defined structure for data exchange. Modbus can transfer data in serial communication such as RS232 and RS485 using the ModbusRTU and Modbus ACSII variants, while ModbusTCP is used for Ethernet networks. Since the inception of the protocol since the 1970s, it has been widely used in automation for PLC networking and fieldbus communication to field devices. The protocol cannot be used in real time applications such as safety and motion, since it has a multilayered datagram according to the Open Systems Interconnection (OSI) model from the ISO/IEC 7498-1 standard, with a deeply buried data payload, slowing down communication.

3.2.3. OPC UA Protocol

The Open Platform Communications Unified Architecture (OPC UA) [24] is a machine to machine communication protocol that is open-source, cross platform and based on the IEC 62541 standard. Two instants can communicate through server/client and publish/subscribe infrastructure. Both RoboDK and CODESYS runtime control can communicate through OPC UA.

3.2.4. Python codes and libraries

Python is a high-level general-purpose programming language, which gained great popularity in the past decade due to coding simplicity, mobility, extensive libraries and community support. There are many communication libraries using the Modbus protocol such as PyCOM and PyModbus, which are imported to the python codes used in RoboDK, to allow RoboDK to communicate to the physical twin or CODESYS runtime control. There is also a simple Python library OPC UA server and client (Python OPCUA) to communicate through the OPC UA protocol.

4. The creation process of the digital twin system

To accommodate the granular nature of the educational process in the automation and systems integration field, this article represents three stages of the digital twin creation process based on connection to the physical twin and data processing. These are Simulation of data and system, Demonstration of past and present data, and finally, Emulation and Learning from data. These successive stages allow the natural progression of the digital twin from simplicity to complexity, giving students and educators better understanding along the way.

4.1. Stage 1: Simulation of data and system

In RoboDK, 3D models of all LEAF modules have been created (Fig. 2). The established digital twin can be used to run different scenarios before importing those later to the physical system. If this is a repeated simulation cycle with an existing physical twin, all data exchanges with that physical twin should be first cutoff, and system layout and initial state should be forced in RoboDK. Sensors and checks have been implemented using Python codes to simulate the behavior of the physical system. The simulated system behavior is only limited to field of automation and systems integration, which is the main point of interest in this paper, therefore, the digital twin only shows the motion of robots, movement of pallets, presence of those pallets at the location of the sensors, etc. and the reaction of controllers such as PLCs and robot controllers.

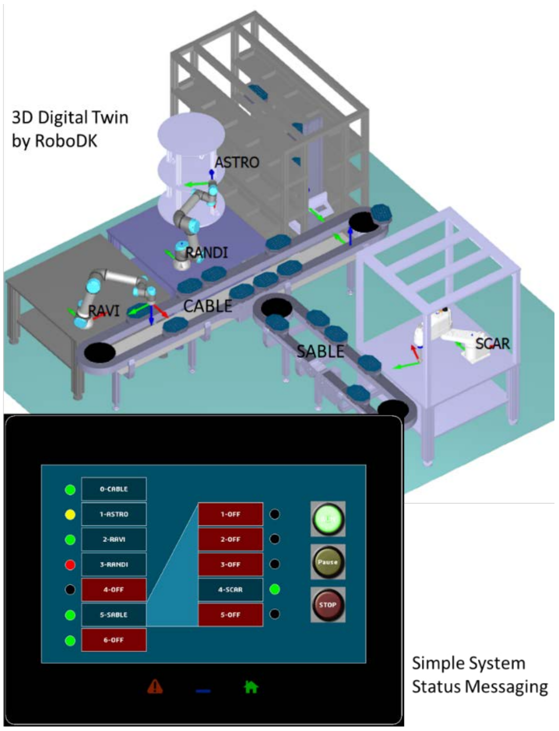

Fig. 3. RoboDK 3D simulation and simple messaging

Runtime data and statistics can be collected and logged or displayed, while a simple messaging system (Fig. 2) was created to allow controlling system status as well. Valuable information about tact time, line balancing and buffer sizing can be obtained and improved.

4.2. Stage 2: System Demonstration (past & present data presentation)

4.2.1. Simplified demonstrator

This simplified system demonstrator Fig. 3 uses a lower level control, in which a Python code was implemented directly in RoboDK to interrogate existing system modules, using the Modbus protocol over both serial (RS232 and RS485 communication converted to USB) and Ethernet. In this case, the computer on which RoboDK resides is connected directly to the PLCs and the controllers in the system. The reconfiguration capability of LEAF requires the system to be self-aware of its constituents and layout, which in turn means that the digital twin should be also aware of that structure. A Python code that is pinging the different modules is responsible for building up a system image. This database of modules and locations is then converted into a 3D layout in RoboDK.

The motion of the robots in the system is simply handled by RoboDK with direct Ethernet connection to these robots and their stored models. Using Python, simple messages, similar to stage 1, can be implemented to show the status of the physical twin, if it is running, pausing, in idle mode, etc. The motion of pallet circulation on conveyor tops has been coded using Python. However, RoboDK needs to know where each pallet is at any time. Since LEAF has an RFID system to track the pallets and the WIP they are carrying, it was possible to communicate that back to RoboDK to represent an accurate animation of pallet circulation.

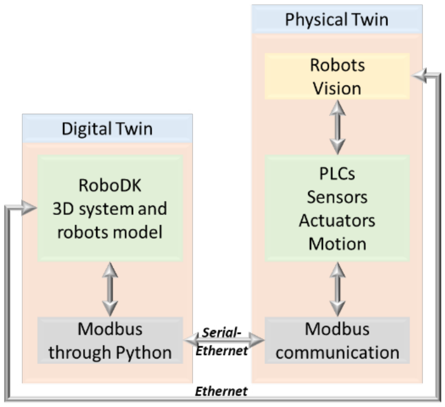

Fig. 2. The simple demonstrator model

4.2.2. Advanced demonstrator

To extend industrial communication to other devices beyond the Modbus protocol, CODESYS 3.5 and its runtime control were used to create a SCADA system that oversees the entire system (Fig. 4). The runtime control has an OPC UA server that was used to broadcast the configuration data to the network, while RoboDK native OPC UA client was used to pull those data to be able to locate the 3D models accordingly.

The CODESYS runtime control SCADA needs to handle the RFID controller data first through ModbusTCP and then publish it to the OPC UA server. During system operation, data logging for historical trends is performed in both the physical system using SCADA and in the digital model in RoboDK by storing logged data in CSV files, that can be then demonstrated and manipulated using Excel sheets.

4.3. Stage 3: emulation and learning from data

Part of the learning process in automation and systems integration takes place in a regular classroom during lecture time. To increase the level of engagement, system emulation can be brought to the classroom, by having a running emulated system controller and data stream on computers (Fig. 5). In the case of LEAF, CODESYS 3.5 runtime control was installed on student laptops to emulate LEAF SCADA. A

Python code was created to generate ModbusTCP tags from RoboDK simulation sensors, which have been created in stage 2. The runtime control then takes the ModbusTCP tags and publish them to the OPC UA server, after which the RoboDK OPC UA client pulls those tags and demonstrates the model accordingly. This phase is a closed loop system simulation-emulation, that starts from a forced system configuration and initial state. Other data can be forced or artificially injected to the runtime control emulated controller to demonstrate the effect on the simulated 3D model.

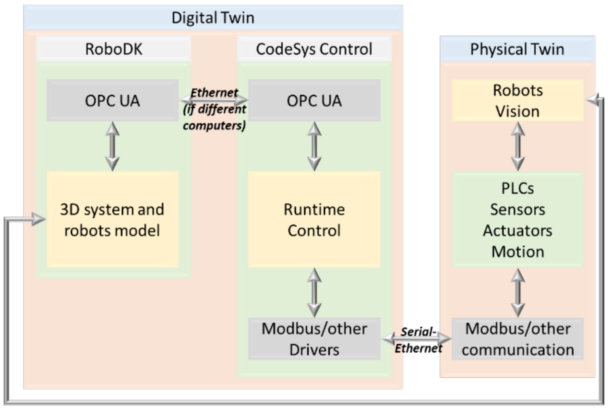

Fig. 4. The advanced demonstrator model

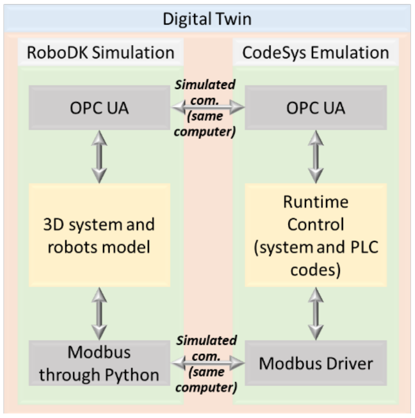
Digital Twin
RoboDK Simulation
CodeSys Emulation
OPC UA
Simulated com. (same computer)
OPC UA
3D system and robots model
Runtime Control (system and PLC codes)
Modbus through Python
Simulated com. (same computer)
Modbus Driver

Fig. 5. The digital twin emulator model

5. Discussion

Learning factories are crucial tools for experiential education in the fields of manufacturing engineering, automation and systems integration. However turnkey learning factories cannot give students and trainees the maximum learning hands-on experience they need, since those systems are similar to black boxes, and most educational institutes don't have the capacity or the knowhow to modify their hardware or software, in addition to being inherently difficult to modify. This paper presented a method to create a digital twin of an open-source low-cost changeable learning factory, since digital twins have gained high importance in industry, while being great simulation and educational tools complementing their physical twins.

The presented digital twin creation model uses a collection of open-source free or low-cost digital components to construct the digital twin. These components are RoboDK as a 3D simulation environment, CODESYS IEC 61131-3 based programming environment for automation control, Modbus ASCII/RTU/TCP and OPC UA for communication and Python for programming.

The presented model creates the digital twin over three stages, based on connection to the physical twin and the method of data processing. It starts with simulation of data and system, where only the 3D RoboDK model is used. Then in second stage, a simple then an advanced demonstrator are created for past and present data, while physical system is connected. Finally, emulation and learning from data is established using the RoboDK 3D simulation in conjunction with the CODESYS runtime control emulator, without connection to the physical system. These successive stages should knowledge and understanding to naturally grow from simplicity to complexity.

6. Future work

The current nature of the simulator and emulator models are deterministic, however, converting them into stochastic models will allow the digital twin to work similar to a discrete-event-simulation model, There are many sources of uncertainty in the system such as existing pallet buffers on conveyor tops, the unbalance of cycle times of the different modules, human interference for loading/unloading/ramping up, etc. Another research direction is to look at the Industry 4.0 aspects, by implementing its guidelines and focus on the data analytics, since there is a constant stream of data obtained from the physical twin.

References

[1] Lamancusa JS, Zayas JL, Soyster AL *et al.* The learning factory: Industry-partnered active learning. Journal of Engineering Education 2008; 97:5-11.
[2] ElMaraghy H, AlGeddawy T, Azab A, ElMaraghy W. Change in Manufacturing – Research and Industrial Challenges. In: 4th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2011). Edited by: ElMaraghy H. Montreal, Canada: Springer; 2011.
[3] Abele E, Chryssolouris G, Sihn W *et al.* Learning factories for future oriented research and education in manufacturing. CIRP Annals - Manufacturing Technology 2017; 66:803–826.
[4] Tisch M, Abele E, Metternich J. The Life Cycle of Learning Factories for Competency Development. In: Learning Factories : Concepts, Guidelines, Best-Practice Examples. Cham: Springer International Publishing; 2019. pp. 127-198.
[5] Wagner U, AlGeddawy T, ElMaraghy H, Müller E. The State-of-the-Art and Prospects of Learning Factories. In: 45th CIRP Conference on Manufacturing Systems (CIRP CMS). University of Patras, Laboratory for Manufacturing Systems and Automation, Athens, Greece: 2012.
[6] Wiendahl H-P, ElMaraghy HA, Nyhuis P *et al.* Changeable Manufacturing: Classification, Design, Operation. Keynote Paper, CIRP Annals 2007; 56:783-809.
[7] Chryssolourisa G, Mavrikiosa D, Rentzosa L. The Teaching Factory: A Manufacturing Education Paradigm. In: 49th CIRP Conference on Manufacturing Systems (CIRP-CMS 2016). Procedia CIRP; 2016. pp. 44 – 48.

[8] Festo-Didactic. iFactory: Innovative training factory. In: www.festo-didactic.com.
[9] Modoni G, Caldarola EG, Sacco M, Terkaj W. Synchronizing physical and digital factory: benefits and technical challenges. Procedia CIRP 2019; 79:472-477.
[10] Kuts V, Modoni GE, Otto T *et al.* Synchronizing physical factory and its digital twin through an IIoT middleware: a case study. Proceedings of the Estonian Academy of Sciences 2019; 68:364-370.
[11] Grinshpun G, Cichon T, Dipika D, Rossmann J. From Virtual Testbeds to Real Lightweight Robots: Development and deployment of control algorithms for soft robots, with particular reference to. In: Proceedings of ISR 2016: 47st International Symposium on Robotics. 2016. pp. 1-7.
[12] Dukalski R, Çençen A, Aschenbrenner D, Verlinden J. Portable Rapid Visual Workflow Simulation Tool for Human Robot Coproduction. Procedia Manufacturing 2017; 11:185-197.
[13] Vila C, Ugarte D, Ríos J, Abellán JV. Project-based collaborative engineering learning to develop Industry 4.0 skills within a PLM framework. Procedia Manufacturing 2017; 13:1269-1276.
[14] Pagan Ms J. STEM Education Using Emulation Software for Hydraulic Fluid Power Applications. Journal of Business and Management Sciences 2020; 6:86-92.
[15] Kousi N, Gkournelos C, Aivaliotis S *et al.* Digital twin for adaptation of robots' behavior in flexible robotic assembly lines. Procedia Manufacturing 2019; 28:121-126.
[16] Buhl JF, Grønhøj R, Jørgensen JK *et al.* A Dual-arm Collaborative Robot System for the Smart Factories of the Future. Procedia Manufacturing 2019; 38:333-340.
[17] Rosenstrauch MJ, Pannen TJ, Krüger J. Human robot collaboration - using kinect v2 for ISO/TS 15066 speed and separation monitoring. Procedia CIRP 2018; 76:183-186.
[18] James S, Freese M, Davison AJ. PyRep: Bringing V-REP to Deep Robot Learning. In: cs.RO. Cornell University arXiv; 2019.
[19] RoboDK. Simulate Robot Applications. In: https://robodk.com/. accessed 2020.
[20] Ribeiro Filipe M, Pires JN, Azar Amin S. Implementation of a robot control architecture for additive manufacturing applications. Industrial Robot: the international journal of robotics research and application 2019; 46:73-82.
[21] AlGeddawy T. A Simplified Changeable Learning Factory Design Based on a Granularity Complexity Model. Procedia Manufacturing 2019; 38:654-662.
[22] 3S-smart. CODESYS: IEC 61131-3 automation software for engineering control systems. In: https://www.codesys.com/. accessed 2020.
[23] Modbus-Organization. Modbus protocol. In: http://www.modbus.org/. accessed 2020.
[24] OPC-Foundation. The Industrial Interoperability Standard: OPC UA. In: https://opcfoundation.org/. accessed 2020.