Accurate Robot Simulation for Industrial Manufacturing Processes using FMI and DCP Standards

Nihar Hasmukhbhai Shah[1] Perig Le Henaff[2] Clemens Schiffer[3] Martin Krammer[3]
Martin Benedikt[3]

[1]AIRBUS Operations GmbH, Germany, nihar.shah@airbus.com
[2]AIRBUS Operations S.A.S., France perig.lehenaff@airbus.com
[3]Virtual Vehicle Research GmbH, Austria,
{clemens.schiffer,martin.krammer,martin.benedikt}@v2c2.at

Abstract

Increased demand for customized products and reduced manufacturing times are key drivers towards modern, automated manufacturing systems. Manufacturing companies increasingly rely on simulation models of their manufacturing systems, with the goal to optimize critical production parameters and programming of their industrial assets. Simulation driven optimization concepts like digital twin and virtual commissioning are gaining popularity among manufacturing units to drive production rates higher. Manufacturing systems in the aerospace domain are highly complex, due to component size, tight tolerance requirements, and multi-tier manufacturing processes. Accurate simulations of robots and other programmable assets are needed, in order to lower the risk of collisions and manufacturing down times. In practice, this leads to inhomogeneous and even proprietary simulation environments, with different software interfaces. In this paper we introduce an accurate robotic arm simulation for industrial manufacturing robots that is based on open standards. This simulation environment is based on two open access standards, namely the Functional Mock-up Interface (FMI) and the Distributed Co-Simulation Protocol (DCP). In a virtualized manufacturing process the number of involved stakeholders is significantly higher. Typically, it includes software and simulation tool vendors, next to the robotic system providers. Therefore a modular software architecture based on open access standards is considered beneficial. Due to the fact that passenger aircraft are highly customized, frequent reprogramming of robotic systems is needed. During these component manufacturing processes the challenge is to maintain a high level of accuracy and reliability.

Keywords: manufacturing, robotics, co-simulation, virtualization, standards

1 Introduction

1.1 Motivation

The digitization of manufacturing progresses towards the paradigm of Industry 4.0 (Lasi et al. 2014). We can observe a strong shift from manufacturing products in a repetitive way to a significantly more smart and intelligent way of manufacturing. Especially in airplane industry, where orders are highly customized, including lots of individual adaptations, following high numbers of variants. Manufacturer Airbus produces parts across seven European countries (Mas et al. 2013) and finally assembles them to complete airplanes. The production of airplanes is massively distributed, and so is the entire supply chain behind manufacturing. Production processes are optimized for concurrency and collaboration. The goal is to enable short time-to-market and reduced cost. At the same time, quality levels should be maintained or even increased.

Today industrial robots are increasingly used in many airplane manufacturing steps. Manual offline programming of manufacturing robots for large aircraft components is a difficult task. There are several reasons for that. First of all, offline programming refers to the process of defining robotic movements when the robot is not in service. The manufacturing line has to be stopped for the programming process. Second, offline programming is constrained by numerous frame conditions. For example, space for robot movement is often limited, due to robot or machinery placement and complexity of parts. Third, inadvertent contact and collisions between the robot and airplane parts must strictly be avoided. Airplane materials and parts are expensive, and even partly manufactured composite parts are valuable. Damaged parts must be replaced, this adds up additional cost, generates waste, and slows down production. The further manufacturing processes are progressed, the more important it is to avoid damage to parts. For these reasons collision free path planning is important. In practice, even small changes to the manufacturing process may have severe impact to robot movement, and therefore programming. Therefore maximum flexibility of configuration and reconfiguration is key to speed up manufacturing and increase facility output. Finally and fourth, manual robot programming is time consuming and subject to improvement.

Due to advancements in simulation technology, virtual validation has been identified as a key method to over-

come aforementioned problems. Virtual validation refers to the solution, where a software programmed robot can be tested and evaluated in a simulated space, before the moving on to the real robot. This work focuses on the infrastructure required for virtual validation. As airplane parts are provided by a large base of suppliers and go through many different process steps, the used software tools are diverse. As a consequence, virtual validation needs to be able to deal with numerous different interfaces. Supporting many interfaces turns out to be costly.

1.2 Approach

In this work we are investigating the capabilities of FMI and DCP standards, to cope with virtual validation of the behaviour of robotic systems. FMI stands for *Functional Mock-up Interface* (Blochwitz et al. 2011). It represents a software standard for co-simulation in several industry sectors. It was proposed to solve the need for interoperability between models, solvers and tools. FMI was developed in the MODELISAR project, starting in 2008. The FMI specification is standardized as a Modelica Association Project (MAP). Its most recent specification version is 2.0.2 which was released in 2020. The FMI specification document defines an interface for model exchange and co-simulation. Today more than 100 software tools support the FMI[1]. The Distributed Co-Simulation Protocol (DCP) is an application-level communication protocol. It was designed to integrate models or real-time systems into simulation environments. It was developed in the ACOSAR project (Krammer, Marko, and Martin Benedikt 2016). It enables exchange of simulation related configuration information and data by use of an underlying transport protocol (such as UDP, TCP, or CAN). At the same time, the DCP supports the integration of tools and real-time systems from different vendors. The DCP is intended to make simulation-based workflows more efficient and reduce the overall system integration effort. It was designed with FMI compatibility in mind, i.e., it follows a master-slave communication principle, uses an aligned state machine implementing an initialization mechanism, and defines an overall integration process which is driven by standardized XML file formats. Version 1.0 of the DCP specification document was released as an open-access Modelica standard in early 2019 (Krammer, Martin Benedikt, et al. 2018; Krammer, Schuch, et al. 2019).

We aim at an accurate simulation of a universal robot (UR10) robotic arm. The UR10 is not compatible with, e.g., the RRS2 (realistic robot simulation) protocol. To overcome this issue, simulation environment provided by the robot vendor shall be used. For that purpose, a DCP master and slave pair shall be embedded into a FMU. The FMU can then be consumed by any FMI compatible robot programming software tool. The UR10 provides a virtual robot controller (VRC) through a virtual machine. Additionally, UR drivers are openly available and adopted by ROS (Robot Operating System). Its interfaces are specified in an open manner, but vendor specific. Due to the open nature of FMI and DCP, the necessary interfaces and configurations can be adapted effectively.

This paper is structured as follows. Section 2 provides an overview of related work from the fields of robotics and distributed co-simulation. In Section 3 our main contribution is stated. Section 4 highlights main results. Section 5 summarizes and concludes this paper.

2 Related Work

Realistic Robot Simulation (RRS) (Bernhardt, Schreck, and Willnow 1994; Bernhard, Schreck, and Willnow 2001; Bernhardt, Schreck, and Willnow 2001; Bernhardt, Schreck, Willnow, and Baumgartner 2002) is an initiative of automotive companies, robot manufacturers, simulator manufacturers, line builders, and measurement system manufacturers. It aims at enhancement of robot simulation accuracy and methodologies for robot off-line programming. RRS-2 defines a Virtual Robot Controller (VRC) interface. Its specification is maintained by Fraunhofer IPK and is not openly available. Only few robot manufacturers are offering VRCs compatible to the RRS-II protocol for simulation purpose. Most manufacturers have their own software solutions for realistic simulation. Unfortunately those are often not compatible to their product life cycle management (PLM) solutions.

A framework based on OPC-UA for distributed industrial robot control is shown in (Vick and Krüger 2018). It aims at virtualization on cloud systems and virtual machines, and uses the UR10 robot. The integration of simulation systems into OPC-UA networks is shown in (Reitz and Rosmann 2020). It focuses on data mapping from a simulation meta data model to an OPC-UA information model. The architecture allows for concurrent message passing between an OPC-UA server and the simulation. This allows for simulation of entire scenarios of an automotive production line.

This publication focuses on co-simulation techniques. A survey regarding the wide field of co-simulation is presented in (Gomes et al. 2018). A real-time co-simulation platform for virtual commissioning of production systems is presented in (Scheifele, Verl, and Riedel 2019). In a similar way, virtual commissioning using co-simulation for virtual plants is shown in (Süß, Strahilov, and Diedrich 2015). A co-simulation platform for design of networked control systems is shown in (W. Li, Zhang, and H. Li 2014).

In the field of co-simulation, the coupling between variables represents one of the largest challenges (Martin Benedikt and Hofer 2015). A solution to overcome the coupling challenge is presented in (Benedikt et al. 2013). In (Stettinger et al. 2014) co-simulation is extended to the real-time domain by using a model-based

[1] http://fmi-standard.org/tools/

DOI 10.3384/ecp21181673

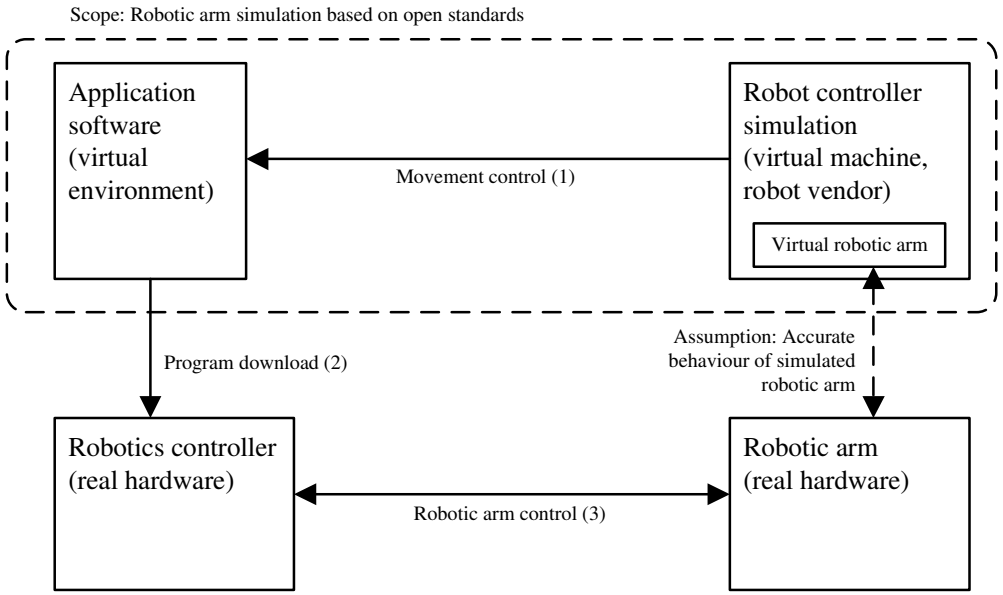

Figure 1. Accurate simulation of a robotic arm system, including simulation based on open standards.

extrapolation scheme, to compensate round-trip time and noise-handling. A mixed real-virtual prototype from the automotive domain based on utilization of the DCP is shown in (Baumann et al. 2019). A distributed demonstrator consisting of a small scale test bed connected to a co-simulation environment is used for performance evaluation. Another example for a real-time co-simulation application can be found in (Rehtanz and Guillaud 2016).

3 Co-Simulation Architecture for Robotic Arm Control

3.1 Concept

The main concept of the proposed solution is shown in Figure 1. The application software on the upper left side provides a virtual environment for the robot and the objects it interacts with. This application software is able to integrate functional mock-up units (FMUs), hence it acts as an FMI master.

The robotics simulation on the top right hand side shall be provided by the robot vendor. This ensures the best possible simulation, having compatibility, consistency, and accuracy in mind. For the intended target robot UR10 a virtual machine is available. This virtual machine is accessible and provides an entry point for adaptations. It is intended to act as a DCP slave.

The center part of the concept is a co-simulation platform. It must be able to control one or more robot simulations, thus act as a DCP master. On the other hand, it must abstract robot connectivity behind the FMI, and act like an FMU for co-simulation.

The goal is to build an accurate and real-time capable robotic arm simulation, with a simulation infrastructure that is fully based on open standards. In a broader sense of virtual validation, the robotic arm simulation shall be used as a central component for safe and reliable robot programming. Finally, the simulation shall imitate the movement and behavior of the real hardware robotic arm, so that manufacturing processes can be planned and analyzed with the best possible predictive capabilities.

3.2 Implementation

The aforementioned concept was implemented as shown in Figure 2. The application software (3DExperience) is capable of running FMUs. To establish communication between the virtual environment and the virtual robot controller a co-simulation platform is used. There are three main characteristics of the co-simulation platform.

1. It can be accessed as an FMU from the outside.
2. It is capable of acting as a DCP master.
3. It is able to provide DCP slave functionality.

The DCP master is capable of configuring and operating a DCP simulation scenario. Technically, it will establish a configuration for two slaves. It distributes configuration information, such as variable input and output configurations. Typical parameters include step size and time resolution, as well as the network configuration information for each variable. In our architecture one slave (slave A) represents the robotic simulation. It is implemented using DCPLib. The other slave (slave B) ensures DCP access to the co-simulation platform. This summarizes the logical view on our architecture.

However, in reality the DCP master and slave B are realized in a monolithic fashion. It is able to send and receive simulation data as defined in the DCP specification.

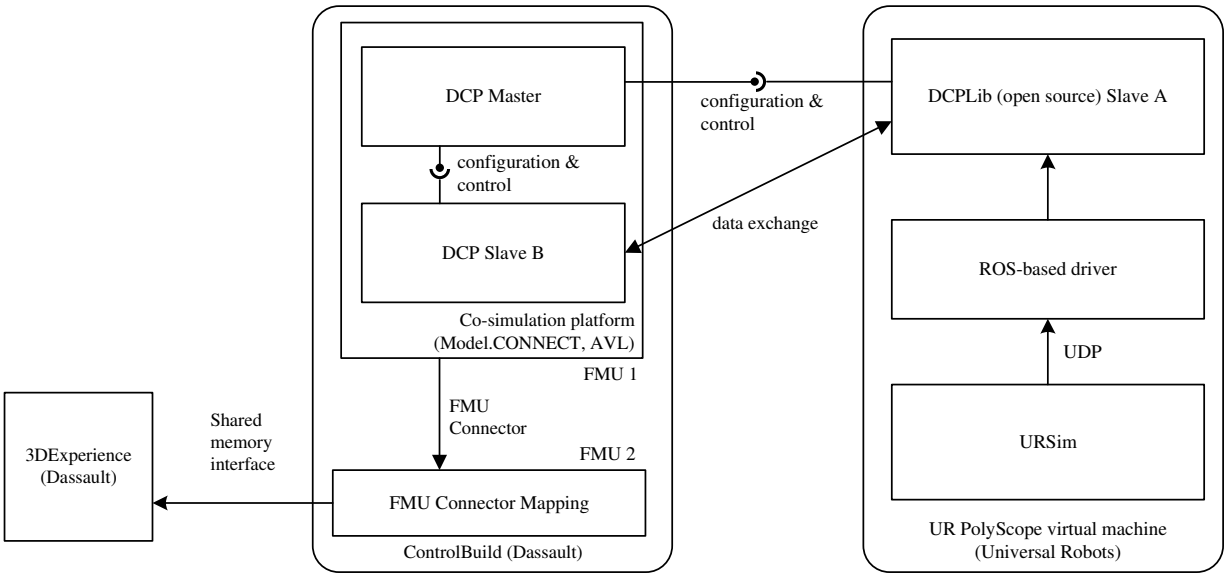

Figure 2. Realized tool couplings and interfaces to use URSim with 3DExperience

This simplifies the architecture as it is the only DCP component in the co-simulation platform FMU. If the used DCP master would not be capable of data exchange, a separate, encapsulated DCP slave (representing slave B) is needed within the FMU for this purpose.

When the FMU is instantiated by the FMI master the DCP scenario is configured and started. This results in data exchange via the DCP protocol between FMU 1 and the robotic controller simulation. This data is then relayed via FMU 2 to the application software (3DExperience) and vice versa.

3.3 Configuration Management

One of the main advantages of this approach is its impact on configuration management. Both FMI and DCP standards rely on static description files. So for each FMU a `modelDescription.xml` contains the necessary information for structural integration. In a similar way the DCP slaves, in particular Slave A, can be structurally integrated by use of a standardized DCP slave description file. The co-simulation platform consumes this DCP slave description file. After that, the co-simulation platform needs a mapping, to associate all variables from DCP to variables of FMI, and vice versa. The only code that needs to be compiled in a build process from source is the DCPLib code for Slave A.

3.4 Scalability

In many industrial manufacturing tasks multiple robots have to cooperate or interact with other manufacturing appliances. Typical examples are multiple robotic arms performing work on the same part, transportation vehicles, conveyor belts, and similar. Following the introduced simulation architecture, a larger number of DCP slaves is required to incorporate these appliances. This can be achieved in two different ways.

Assuming that one instance of a co-simulation platform is used to control one DCP slave, the application software must be capable of integrating multiple FMUs. By using this solution, the complexity of DCP communication remains at a low level. But at the same time, multiple instances of the co-simulation platform are required. This poses increased requirements to resources (memory, CPU, etc.) of the host system(s).

In contrast to this solution, the application software may continue to use one single FMU, but increase the number of exchanged variables to control more devices. This can be achieved by using array data types or multiple variables. In this case, the co-simulation platform has to register and control multiple DCP slaves. Technically, there are two options for this as well. Instantiation of one master per DCP slave, or instantiation of one single master controlling multiple slaves.

Combinations of these two possibilities are feasible. In any way, the proposed architecture is considered to be scalable, which also depends strongly on the underlying hardware resources and their configuration.

3.5 Time Regime

The virtual robot controller has an operating frequency of 125Hz. Therefore, the operating mode of the DCP scenario was set to SRT (soft real-time) (Krammer, Mar-

DOI 10.3384/ecp21181673

tin Benedikt, et al. 2018). This means that the absolute time should be synchronized with the simulated time. The DCP output step size of the virtual robot controller is configured with 0.008ms. The DCP step size is handled independently of the FMI step size. It is guaranteed that a FMI do-step call yields the most recent DCP variable value. 3DExperience acts as FMI importer. For real-time execution of the entire simulation scenario, the FMI importer must periodically call the do-step function with the same frequency and step size as used by the DCP master.

3.6 Software Tools

DCPLib is an open-source software library maintained by Modelica Association Project (MAP) DCP. Its initial version was created as a deliverable during the ITEA 3 ACOSAR project. It consists of several packages. The core library contains common classes, like constants and PDU definitions. Furthermore, master and slave packages are available, to rapidly create DCP slaves and a master to control them. DCPLib supports UDP and TCP over IPv4 transport protocols. Furthermore, it includes packages for generation and processing of ZIP- and XML-based description files. DCPLib was used for the robotics simulation part of this work.

Model.CONNECT™ is an open model integration and co-simulation platform from AVL GmbH. Typically, it improves development efficiency by interlinking simulation models into a consistent virtual prototype. The origin of simulation models is arbitrary, as the architecture of Model.CONNECT™supports the integration of up to 40 different modeling tools. Next to these well-known modeling and simulation tools, Model.CONNECT™supports a number of open standards, including Modelica Association's FMI, DCP, and SSP specifications. This allows for pure virtual and also mixed real-virtual prototypes, based on open standards. Model.CONNECT™was used as a co-simulation platform because it already supports all necessary interfaces (like acting as a DCP-master for DCP-slaves) and a Model.CONNECT™model can be exported as an FMU to allow the interaction with any tool that can import Co-Simulation FMUs.

ControlBuild is a software platform by Dassault Systemes. ControlBuild is an open automation software platform that allows seamless progress through all phases of the application development cycle – from definition and validation of specification to implementation and deployment (Systemes n.d.). ControlBuild Validation allows virtualization of physical industrial installations. A large part of the tests is traditionally carried out on-site. In the following integration phase tests are simulated on a test platform in a near-real-life environment. ControlBuild is part of the 3DExperience portfolio and provides seamless interfaces with the 3DExperience platform for simulation and validation purposes. Based on a model-driven approach and supported by a structured set of libraries, ControlBuild is used to efficiently model, simulate, test, validate and deploy control applications according to IEC61131-3 (Programmable controllers - Part 3: Programming languages). Furthermore, ControlBuild allows co-simulation of virtualized models of different industrial assets. It is able to link their behaviour to corresponding digital representations in 3DExperience through standardized interfaces, such as FMI.

3DExperience 3DExperience is a cloud-based collaborative Product Lifecycle Management platform by Dassault Systemes. It contains software solutions for all phases of the product life cycle supporting the digital design and development of products that are subsequently manufactured. As a platform, it houses multiple capabilities (applications or apps) in a single seamless piece of software. DELMIA (Digital Enterprise Lean Manufacturing Interactive Application) is part of the 3DExperience platform that provides specialized solutions for digital manufacturing and simulations. 3DExperince supports standardized simulation interfaces, such as FMI.

4 Results

The introduced concept was implemented, including the described tool couplings and interfaces. Figure 3 shows a screen capture of the running robot simulation. On the upper side of the image the debugging output of DCPLib can be seen. `DAT_input_output` PDUs are sent, they contain a payload carrying `float64` data type values. In the rolled out DCP configuration one `data_id` per variable was used. On the lower side of the image the graphical programming environment of UR10 can be seen. It shows robotic arm controls next to the status of all joints of the robotic arm.

Figure 4 shows a visualization of a sequence of movements of the robotic arm in a three-dimensional space. The trajectories were plotted by the robot's tool tip. The dashed line represents the movement trajectory as calculated by 3DExperience (3DX). It shows clear and idealized characteristics. In contrast to that, the continuous line represents the movement trajectory as calculated by the virtual robot controller (VRC). This trajectory shows some deviations compared to the trajectory of 3DExperience.

The difference between both trajectories was calculated by using a minimum-distance algorithm. For one trajectory point, the minimum distance to a line defined by the two closest points of the other trajectory was calculated. As a result the maximum deviation of the virtual robot controller's path from the 3DExperience's path was 2.793 millimeters. The standard deviation across the entire sequence of movements amounts to 0.421 millimeters, the mean value was determined to be 0.691 millimeters. URSim considers additional robotic parameters, as physical structures and materials, as well as mechanic joints and

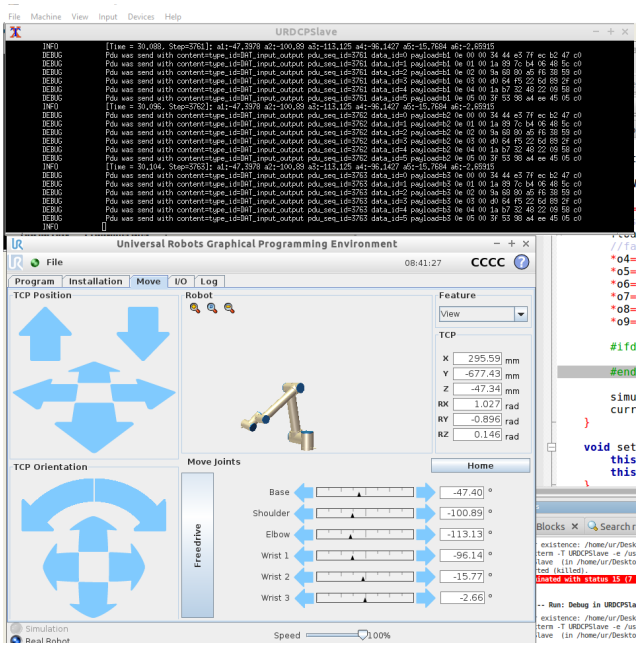

Figure 3. Universal Robot UR10 Simulation as DCP Slave A

hinges. It is expected that this leads to a more accurate simulation result, which is considered being closer to reality as the reference trajectory by 3DExperience.

5 Conclusion

In this paper we highlight a flexible concept for accurate simulation of a robotic arm. Its simulation architecture is based on open-access standards. Furthermore, it relies on open-access interface specifications and partly on open-source software.

We have successfully demonstrated the feasibility of our approach. The proposed simulation architecture is highly modular, mainly due to the use of FMI and DCP. These open-access interface and protocol specifications provide the most flexibility. This is not only true for large original equipment manufacturers, but the entire approach has the potential to (re-)align the entire supply chain of industrial robotics and manufacturing. The proposed approach poses a strong shift from custom software tools to configurable, modular, special purpose software tools for robotics and manufacturing. The added value originates from the capability to configure the involved software tools and their interfaces. The efforts spent for coding were reduced to a minimum. In our case, DCPLib was the only software package that required a build process.

Future work includes the process of scaling up the introduced solutions to full manufacturing processes or parts thereof. For example, multiple robots operating in parallel require respective collaborative solutions for simulation. This can be achieved by modification of interfaces, and parallelization of communication to distributed components. Finally, a set of virtual robots could be effectively created by multiple instantiation mechanisms.

References

Baumann, Peter et al. (2019). “Using the Distributed Co-Simulation Protocol for a Mixed Real-Virtual Prototype”. In: *Proceedings - 2019 IEEE International Conference on Mechatronics, ICM 2019*. Ilmenau, Germany: IEEE Industrial Electronics Society, pp. 440–445. ISBN: 9781538669594. DOI: 10.1109/ICMECH.2019.8722844.

Benedikt, M et al. (2013). “NEPCE-A nearly energy-preserving coupling element for weak-coupled problems and co-simulations”. In: *Computational Methods for Coupled Problems in Science and Engineering V - A Conference Celebrating the 60th Birthday of Eugenio Onate, COUPLED PROBLEMS 2013*, pp. 1021–1032. ISBN: 9788494140761.

Benedikt, Martin and Anton Hofer (2015). “Guidelines for the application of a coupling method for non-iterative co-simulation”. In: *Proceedings - 8th EUROSIM Congress on Modelling and Simulation, EUROSIM 2013*, pp. 244–249. ISBN: 9780769550732. DOI: 10.1109/EUROSIM.2013.52. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7004951.

Bernhard, Rolf, Gerhard Schreck, and Cornelius Willnow (2001). “DEVELOPMENT OF VIRTUAL ROBOT CONTROLLERS AND FUTURE TRENDS”. In: *6th IFAC Symposium on “Cost oriented Automation”*.

Bernhardt, Rolf, Gerhard Schreck, and Cornelius Willnow (1994). “The Realistic Robot Simulation (RRS) Interface”. In: *IFAC Proceedings Volumes* 27.4, pp. 321–324. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)46044-7. URL: http://dx.doi.org/10.1016/S1474-6670(17)46044-7.

Bernhardt, Rolf, Gerhard Schreck, and Cornelius Willnow (2001). “Virtual Robot Controllers as Simulation Agents”. In: *Workshop on Agent-Based Simulation, SCS - The Society for Modeling and Simulation International in cooperation with ASIM - Arbeitsgemeinschaft Simulation*, pp. 1–6.

Bernhardt, Rolf, Gerhard Schreck, Cornelius Willnow, and Alan Baumgartner (2002). “Realistic Robot Simulation in Concurrent Engineering of Manufacturing Lines in Automotive Industries”. In: *Eighth ISPE INTERNATIONAL CONFERENCE ON CONCURRENT ENGINEERING : RESEARCH AND APPLICATIONS*.

Blochwitz, Torsten et al. (2011-03). “The Functional Mockup Interface for Tool independent Exchange of Simulation Models”. In: *In Proceedings of the 8th International Modelica Conference*, pp. 105–114. ISBN: 978-91-7393-096-3. DOI: 10.3384/ecp11063105.

Gomes, Cláudio et al. (2018). “Co-Simulation: A Survey”. In: *ACM Computing Surveys* 51.3, pp. 1–33. ISSN: 0360-0300. DOI: 10.1145/3179993.

Krammer, Martin, Martin Benedikt, et al. (2018). “The distributed co-simulation protocol for the integration of real-time systems and simulation environments”. In: *Simulation Series*. Vol. 50. 10, pp. 1–14. ISBN: 9781510860230. DOI: 10.22360/summersim.2018.scsc.001. URL: https://dl.acm.org/citation.cfm?id=3275383.

Krammer, Martin, Nadja Marko, and Martin Benedikt (2016). “Interfacing Real-Time Systems for Advanced Co-Simulation - The ACOSAR Approach”. In: *STAF 2016 Doctoral Symposium and Projects Showcase*. Ed. by Catherine Dubois et al. Vienna, Austria, pp. 32–39.

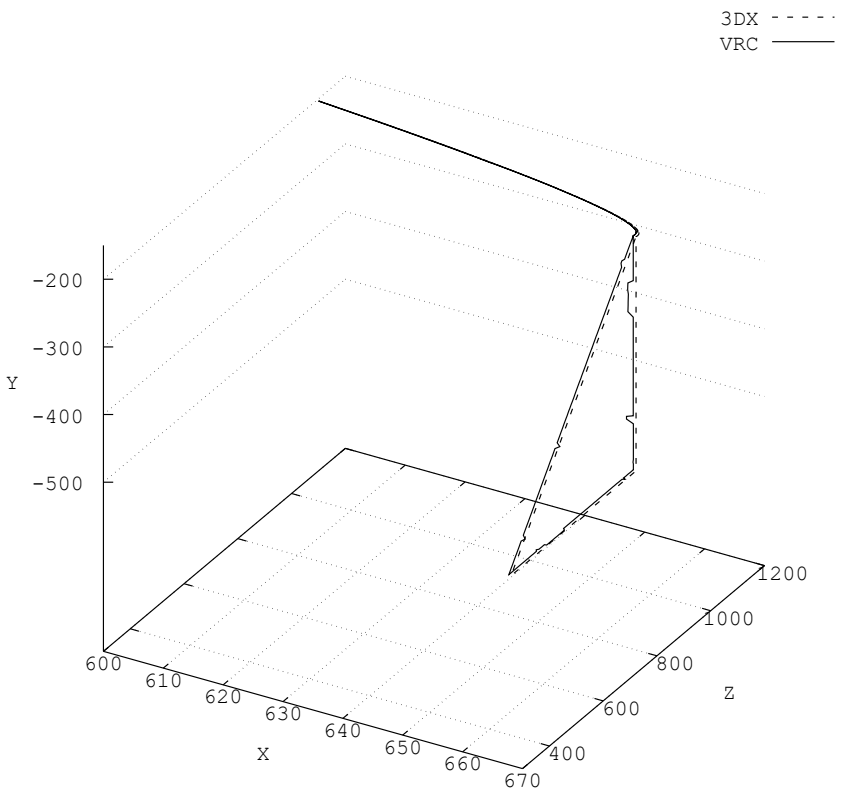

Figure 4. Visual comparison of robotic arm movement trajectories, as recorded with 3DExperience and Virtual Robot Controller

Krammer, Martin, Klaus Schuch, et al. (2019-02). "Standardized Integration of Real-Time and Non-Real-Time Systems: The Distributed Co-Simulation Protocol". In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. Vol. 157. Linköping University Electronic Press, Linköpings universitet, pp. 87–96. DOI: 10.3384/ecp1915787. URL: http://www.ep.liu.se/ecp/article.asp?issue=157%7B%5C%%7D26article=9.

Lasi, Heiner et al. (2014). "Industry 4.0". In: *Business and Information Systems Engineering* 6.4, pp. 239–242. ISSN: 18670202. DOI: 10.1007/s12599-014-0334-4.

Li, Weilin, Xiaobin Zhang, and Huimin Li (2014). "Co-simulation platforms for co-design of networked control systems: An overview". In: *Control Engineering Practice* 23.1, pp. 44–56. ISSN: 09670661. DOI: 10.1016/j.conengprac.2013.10.010. URL: http://dx.doi.org/10.1016/j.conengprac.2013.10.010.

Mas, F. et al. (2013). "Collaborative engineering: An airbus case study". In: *Procedia Engineering* 63, pp. 336–345. ISSN: 18777058. DOI: 10.1016/j.proeng.2013.08.180.

Rehtanz, Christian and Xavier Guillaud (2016). "Real-Time and Co-Simulations for the Development of Power System Monitoring , Control and Protection". In: *Power Systems Computation Conference (PSCC) 2016*. DOI: 10.1109/PSCC.2016.7541030.

Reitz, Jan and Jurgen Rosmann (2020). "Automatic Integration of Simulated Systems into OPC UA Networks". In: *IEEE International Conference on Automation Science and Engineering* 2020-August, pp. 697–702. ISSN: 21618089. DOI: 10.1109/CASE48305.2020.9216827.

Scheifele, Christian, Alexander Verl, and Oliver Riedel (2019). "Real-time co-simulation for the virtual commissioning of production systems". In: *Procedia CIRP* 79, pp. 397–402. ISSN: 22128271. DOI: 10.1016/j.procir.2019.02.104. URL: https://doi.org/10.1016/j.procir.2019.02.104.

Stettinger, Georg et al. (2014). "Model-based coupling approach for non-iterative real-time co-simulation". In: *2014 European Control Conference, ECC 2014*, pp. 2084–2089. ISBN: 9783952426913. DOI: 10.1109/ECC.2014.6862242.

Süß, Sebastian, Anton Strahilov, and Christian Diedrich (2015). "Behaviour simulation for Virtual Commissioning using co-simulation". In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2015-October. ISSN: 19460759. DOI: 10.1109/ETFA.2015.7301427.

Systemes, Dassault (n.d.). *ControlBuild: Designing Automation and Embedded Control Systems*. online, www.3ds.com. accessed on 25 April 2021.

Vick, Axel and Jörg Krüger (2018). "Using OPC UA for distributed industrial robot control". In: *50th International Symposium on Robotics, ISR 2018*, p. 501.