Towards Digital Twins for Industrial Assembly - Improving Robot Solutions by Intuitive User Guidance and Robot Programming

Sørensen, Lars Carøe; Mathiesen, Simon; Waspe, Ralf; Schlette, Christian

Published in:
Proceedings - 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA

DOI:
10.1109/ETFA46521.2020.9212072

Publication date:
2020

Document version:
Final published version

Towards Digital Twins for Industrial Assembly - Improving Robot Solutions by Intuitive User Guidance and Robot Programming

Lars Carøe Sørensen[1] Simon Mathiesen[1] Ralf Waspe[1] Christian Schlette[1]

***Abstract*— Simulation of robotic tasks allows for cheap evaluation and process optimization, which can then be transferred to the physical system. To avoid discrepancies between physical execution and simulation, real-world process data can be fed back to the simulation environment, a concept referred to as a "Digital Twin". This requires the development of a software architecture, that supports Digital Twins for robot tasks.**

In this work, we propose a system where an operator can take apart a complex assembly, thus creating digitized assembly instructions. These instructions are then used to visually program the robot setup by blocks, which contain functionality ranging from point-to-point motions to high-level skills. These "Skillblocks" allow for a seamless transition between execution in the simulation environment and on the physical robot through interchangeable execution layers in the software architecture. The system also allows for feedback from a physical execution to be monitored in real-time and fed back to the simulation environment for processing.

The aim of the system is to close the gap between digital and physical workcells when integrating robot solutions. This increases intuitiveness and allows for process monitoring and optimization through direct feedback to the digital model.

I. Introduction

Easy programming of industrial robot tasks is of general interest, as robotic automation is introduced in more and more areas of industry. However, programming new robotic tasks are rarely trivial as an integrator must create functionality by bridging a multitude of software interfaces associated with their corresponding hardware components. Furthermore, the integrator has to deal with the complexity of the task the robot actually needs to perform. This can be simple tasks such as a pick-and-place operation, but also more complex robot operations are often needed, such as force-based insertion, polishing, screwing in bolts, etc., which may be supported directly through the robot, or have to be implemented by the integrator. The work described in this paper positions itself within the domain of industrial automated assembly, which adds even further to the complexity by typically including many parts that need to be assembled in a specific order and through several different robot operations. This paper proposes the use of robotics centered Visual Programming (VP) to reduce the complexity of task structuring and robot control. The VP approach presented in this work is structured as blocks of self-contained functionality, denoted as *SkillBlocks*. A SkillBlock encapsulates a robot operation (although this can be extended to other types of hardware) and is represented as a visual programming block of which a network can be created by dragging connections between the blocks, thus creating the sequence operations that make up the assembly task. A visualization of our SkillBlocks and a robot workcell are shown in Fig. 1. The overall goal is to create a system with a suitable architecture and the necessary components to enable a workflow powered by digital models and simulation where the integration of new robotic tasks is facilitated by:

1) Virtual disassembly of production models for creating a sequence, which can then be inverted to create assembly instructions.
2) Automatic generation of networks of SkillBlocks from assembly instructions.
3) Simulation assisted adaption of SkillBlocks to meet user requirements.
4) Continuous process optimization based on feedback from the physical robot to the digital twin.

This work distinguishes itself from previous work, such as [1], by focusing on VP for robot tasks in a scope, where a modular software architecture through an exchangeable layer containing interfaces denoted *Execution Engines* allows for seamless switching between running the programmed task in simulation and on the physical setup. Another novelty compared to [1], is how this creates the possibility of directly connecting the simulated world and the physical setup, enabling a proper *Digital Twin* to be created at the robot cell level. The benefits of this are two-fold: 1) It provides the opportunity to reduce the need for performing tiresome adjustments to the task on the physical platform, as results from learning in simulation can be directly applied on the real robot, and 2) data from the real setup can be fed back into the simulation system and used for process optimization and quality control truly supporting the I4.0 manufacturing principles.

The paper is structured as follows: Section II first examines related work about visual programming and how robotics skills currently are structured. Section III elaborates on our SkillBlocks framework and provide a more concrete explanation of how SkillBlocks are used in connection with the Digital Twin setup. Section IV presents an experiment which uses our SkillBlocks framework in both simulation as well as real-world. In Section V we conclude on this paper and, afterwards, presents future work in Section VI.

[1]SDU Robotics, Maersk McKinney Moller Institute, University of Southern Denmark, Campusvej 55, 5230 Odense, Denmark, {lcs,simat,raw,chsch}@mmmi.sdu.dk

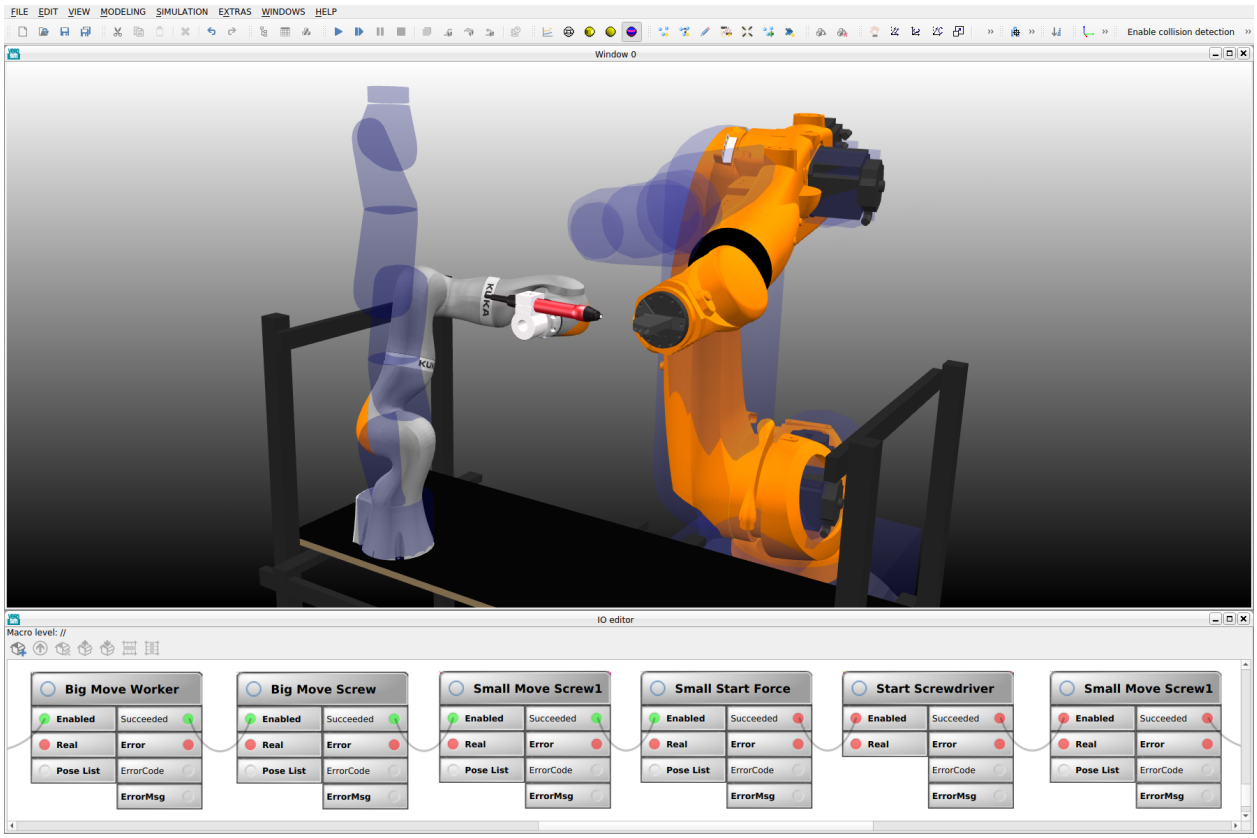

Fig. 1. A scene showing an experimental setup with two robots. The smaller robot (left) are screwing in bolts into a fixture held by the larger robot (right). The full models represent the simulated robots and while the blue silhouettes represent the physical robots in their current configuration. The bottom panel of the figure shows the editor where the SkillBlocks are visualized. SkillBlocks can be chained to execute a sequence of operations.

II. Related Work

VP can be a powerful tool for programming robot control as it provides key features for aiding the programmer [2]:

1) *Concreteness*: Every block of in the network describes concrete functionality independent from any other part of the program.
2) *Directness*: A change to the input of a SkillBlock happens directly on the specific visual representation of it. Furthermore, moving the block and changing its connections directly impacts when that piece of functionality is executed.
3) *Explicitness*: The program describing the industrial task is explicitly visible as a network of blocks and connections that define the order of execution.
4) *Immediate Visual Feedback*: A change to the programmed task can immediately be visualized by running the simulation.

Visual programming is in our opinion a suitable method for the expression of a robot (or another hardware component) skill and thus the conception of SkillBlocks. Prior to this section, we have refrained from the direct use of the term *skill*, as the concept of skills is not strictly defined [3]. Apart from providing a comprehensive review of skill-related literature, the authors of [3] introduce their concept of skills as "*intuitive object-centered robot abilities, which can easily be parameterized by a non-expert*". This notion of being centered around objects springs from their implementation where kinesthetic teaching and gesture-based inputs are used to teach in a robot task in the physical world. Instead, we propose a system based on digital models and co-simulation where frames and coordinates are easier to convey intuitively to a user. These are necessary components to efficiently make use of the data that the manufacturers of the future must possess, as the production industry transitions more and more towards full digitization. Furthermore, the implementation of our SkillBlocks allows for grouping them dynamically at run-time and expose only relevant input and outputs to a non-expert user. A SkillBlock, and the skill it represents, can, therefore, vary in functionality from what the literature typically dub as *Skill Primitives* [3], [4], [5] to high-level functionality as a peg-in-hole operation. Thus, we refuse to adhere to a strict definition of skills for our SkillBlocks as the dynamic nature of the visual blocks allows redefinition of functionality simply by changing connections between SkillBlocks. However, it is clear that the concept of skills as building blocks of self-contained functionality is necessary to facilitate robot programming and thus the system should naturally include basic blocks of the most common skills and support storing and reusing new user-generated skills. The reuse of skills is discussed in [5], which presents a taxonomy for modeling assembly tasks. The authors argue

that tasks modeled using this approach provide flexibility through independence from specific hardware. In our work, this concept is embodied through the use of the previously mentioned Execution Engines. As the network of SkillBlocks can persist even though an Execution Engine is swapped to match another robot or interface. As a first step, the current implementation includes a ROS Execution Engines due to its widespread use in the robotics community.

The authors of [5] presents a software architecture aimed at facilitating easy programming of robotics systems by two key components. First, the system is developed with a strong focus on intuitive use for the benefit of the end-user, and, secondly, the architecture allows hardware-independent skills like in our framework. The major difference to our framework is that system presented in [5] is centered around physical setup with little or no use of simulation and visualization. Our aim is to exploit the latter two components among other while still maintaining a close connection to the physical setup to utilize the Digital Twin concept.

III. The SkillBlocks Framework

The aim of our *SkillBlocks* framework is to facilitate easy programming of robotics tasks though visual programming. The novelty of our framework is the decoupling of functionality, data, and visualization but also the implementation of a Digital Twin concept. The separation of responsibilities minimizes the overhead when implementing new and maintaining existing software. The functionality is in our framework formalized by *Skills* which implements any dedicated functionality. An example of a Skill could be robot motions in joint space. Each Skill has an associated data container concept called *Property Boards*, in which parameters like target configuration and joint speeds for robot motion can be passed.

A. The Digital Twin concept

Having a Digital Twin forms the basis of our SkillBlocks framework. The main idea is to have a full digital copy of the physical workcell on which the same operations can be carried out as if it was in the real-world. Moreover, the Digital Twin should also reflect the physical setup. As an example, a robot would in the digital workcell be manifested using two separate visualizations of that robot representing the state of the robot both in simulation and real-world. This is shown in Fig. 1 where the blue silhouette of each robot represents the state of the physical robot. In this way, a robot configuration can easily be fetched from the physical robot, and then used as a starting point for path planning in simulation which enables visualization and inspection before the chosen path is executed on the physical robot. Hence, a Digital Twin enables a close connection between the simulated and physical workcell, which enables useful features such as simulation of the entire process before real-world execution, easy programming by teaching the physical robot, interaction with the worker doing operation, and monitoring of the process for quality assurance.

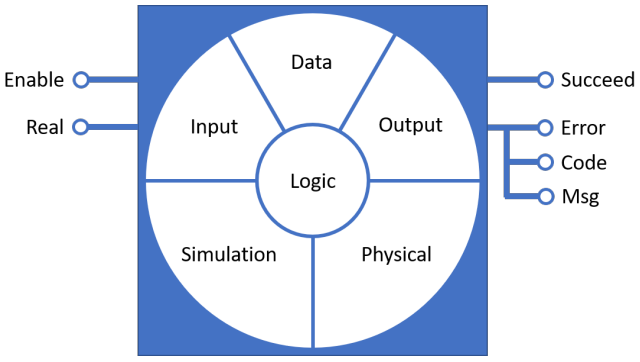

Fig. 2. The structure of a SkillBlock.

B. The Structure of a SkillBlock

Our SkillBlocks framework defines a consistent structure that streamlines the workflow of any SkillBlock. This makes all SkillBlocks act similar which is beneficial for the user experience. A SkillBlock acts in its essence as a bridge between the visual programming and the Skill which contains the actual functionality.

A SkillBlock is initialized with a Skill from which the framework automatically assigns the SkillBlock with a Property Board and an IO-board. The *Property Board* holds the actual data value of the properties of the Skill such as velocities and accelerations for a robot motion. Hence, a Skill can be seen as a generic component since it functionality is separated from the actual input parameters. The *IO-board* of the SkillBlock contains the inputs and outputs of that are visualized and used for connections when building the SkillBlock network. Skills have two *Execution Engines*, which execute the desired functionality in simulation or on the physical setup. In this way, a Skill can be reused for other SkillBlocks without them interfering with each-other's individual internal states and specific input parameters. This minimizes overhead both for implementation of new skills but also for using the software. The same is true for the Execution Engines. Furthermore, the Execution Engines are easily exchanged on the Skill which helps to make our SkillBlocks framework more agile as changes in the physical setup can easily be adapted in the Digital Twin.

The workflow in a SkillBlock is elaborated in the following with support from Fig. 2. At the core of a SkillBlock is the *logic*. The logic reads all *inputs* upon activation. Each SkillBlock also has a Real input which defines if the functionality of the Skill is executed in simulation or on the physical setup by corresponding Execution Engines. Beside the Enable and Real inputs, a SkillBlock can also have additional inputs such as a path (e.g. a list of joint configurations) for a robot motion. The logic handles the necessary preparation before the actual execution. This includes bringing together data from its inputs and property board, but could also be to setup a simulated robot motion with a chosen path, velocities and accelerations. The logic has the possibility to access databases for data exchange prior to the actual execution. Hereafter the execution is carried

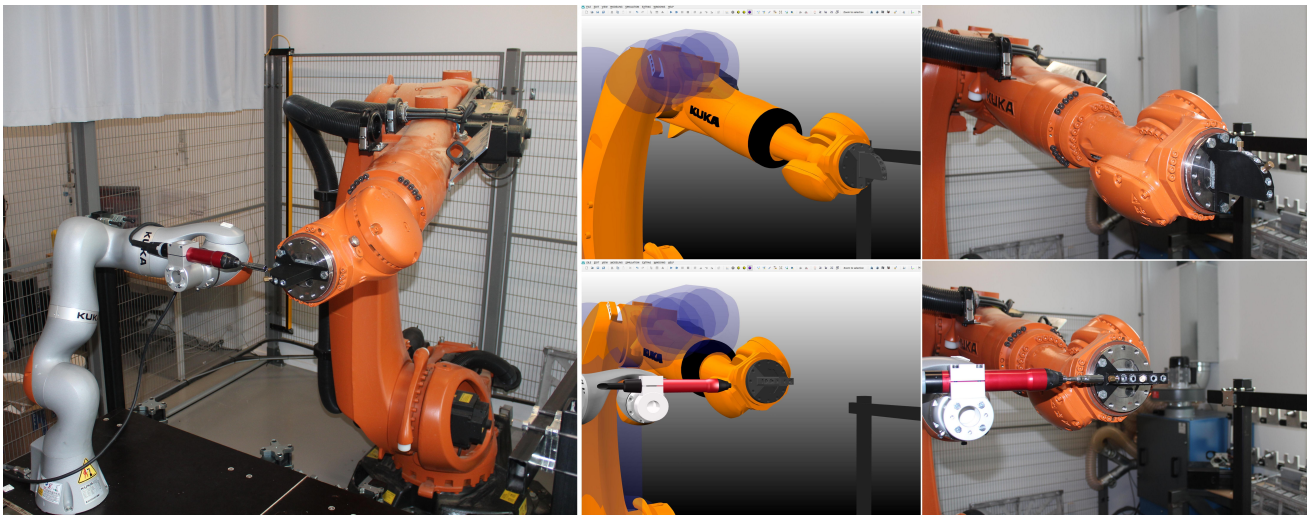

Fig. 3. The physical set up (left) and close-ups of step three and five in both simulation (middle) and real-world (right).

out though one of the execution engine in either simulation or real-world. The logic evaluates wherever the execution performed as expected based on the available feedback from the either simulation or real-world. The evaluation can happen in parallel of the execution or when the execution is finished. Then the logic of SkillBlock can again access relevant databases e.g. to store paths if the purpose of the SkillBlock was path planning. Finally, the outputs are set accordingly to the evaluation.

C. Chaining of SkillBlocks

Examples of SkillBlocks could be, but are not limited to, robot motions (such as point-to-point, linear, and velocity), gripper actions (open and close), integration of functionality from other hardware or software components, or access to databases for saving and loading e.g. robot configurations and paths as well as single parameters.

As previously mentioned it is possible to connect a sequence of SkillBlocks by chaining them together to obtain a more complex behavior. The chaining is done by connecting the Succeeded output of one SkillBlock to the Enable input of another SkillBlock as shown on Fig. 1. In general, an output can be connected to multiple inputs, but an input can only be connected to one output[1]. The sequence of SkillBlocks is activated by the first SkillBlock, and SkillBlocks can work in series, but also in parallel such as opening a gripper while the robot is moving. A sequence of SkillBlocks can furthermore be collapsed into a single new SkillBlock with the same appearance as any other SkillBlocks.

IV. Experiment and Result

This section presents a test case for which our SkillBlocks framework has been applied. The Digital Twin shown in Fig. 1 is used for programming and testing a sequence of robot actions before the exact same sequence is executed on the physical set-up. In the test case, several bolts should be screwed into a demonstration fixture. The fixture is mounted on the larger Kuka KR120 R2500 robot, and a automatic screw driver is mounted on the smaller LBR iiwa 7 R800 robot. The steps in the sequence are as follows: 1) Both robots start in their initial position. 2) The larger robot presents fixture for a worker. 3) The worker inserts two screws without tightening. 4) The larger robot presents the fixture for the smaller robot. 5) The smaller robot performs the necessary screwing operations by using its in-build force controller and its attached screwdriver. 6) Both robots return their initial position. A sequence of SkillBlocks representing the robot operations required for this task is shown in Fig. 1.

Fig. 3 shows the physical setup and close-ups of step three and five in both simulation and real-world. The figure is intended to show that we by our SkillBlocks framework are able to make a Digital Twin of the physical setup. Moreover, a sequence of robotics operations can be constructed by SkillBlocks and then visualized on Digital Twin through simulations prior to the actual execution of the exact same sequence on the physical setup.

V. Conclusion

This paper proposes a new software tool for programming robotics tasks. Centered around the use of a Digital Twin, the programming of robots and additional equipment are done though visual programming using SkillBlocks. Our SkillBlocks framework enables a seamless change between first executing the programmed sequence in simulation hereafter on the physical setup. The main novelty of our software is that SkillBlocks framework decouples functionality, data, and visualization. Moreover, the framework integrates the current status of robots and equipment into the visual representation of the workcell, this with the purpose of easing the programming task.

[1]However, logic operations are available to join signals, such as AND and OR blocks.

VI. Future Work

This paper describes the main components of our SkillBlocks framework including the concept of having a digital twin and how visual programming is utilized. This opens a wide range possible of future work, and in this section we describe the most relevant topics to exploit the possibilities in our framework.

The expansion of the library of Skills remains one of the most urgent topics. This library should contain a wide range of basic robot Skills such as movements etc., but it should also be possible to acquire and store more complex Skills consisting of a sequence of chained SkillBlocks.

Automatic generation and chaining of SkillBlocks will provide the user with a finalized program or at least a suggestion on how to program the robotics setup. The generation should be based on a sequence of assembly steps defined by the user and its creation could be aided by virtual reality.

Fully using the Digital Twin for process monitoring, quality assurance, and supporting the worker are additional topic of relevance in the domain of collaborative industrial assembly and should be addressed in future work.

Finally, extending the hardware interface to support modern interfaces such as the Universal Robot RTDE interface and the Kuka FRI interface is an important task which will enable our software to be used for tasks that require real-time control of the robots.

ACKNOWLEDGMENT

This work was supported by Innovation Fund Denmark as a part of the project "MADE Digital".

References

[1] C. Schlette, D. Losch, and J. Rossmann, "A visual programming framework for complex robotic systems in micro-optical assembly," in *ISR/Robotik 2014; 41st International Symposium on Robotics*. VDE, 2014, pp. 1–6.

[2] M. Burnett, "Software engineering for visual programming languages," in *Handbook of Software Engineering and Knowledge Engineering*. World Scientific Publishing Company, 2001, vol. 2.

[3] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[4] R. H. Andersen, L. Dalgaard, A. B. Beck, and J. Hallam, "An architecture for efficient reuse in flexible production scenarios," in *2015 IEEE International conference on automation science and engineering (CASE)*, 2015, pp. 151–157.

[5] J. O. Huckaby and H. I. Christensen, "A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.