Title:	A Reconfigurable robot workCell for fast set-up of automated assembly processes in SMEs
Acronym:	**ReconCell**
Type of Action:	Innovation Action
Contract Number:	680431
Starting Date:	1-11-2015
Ending Date:	31-10-2018

Deliverable Number:	D4.3
Deliverable Title:	Workcell simulation for reconfiguration
Type (Public, Restricted, Confidential):	PU
Authors :	M. Priggemeyer, D. Kaufmann, and J. Roßmann
Contributing Partners:	MMI

Estimated Date of Delivery to the EC:	31-10-2017
Actual Date of Delivery to the EC:	15-11-2017

Contents

1 Executive Summary

This deliverable outlines the tools for the workcell simulation for reconfiguration. The ReconCell project's high ambitions to allow non-expert users to program the workcell required to further improve the usability of the software tools. During the project review, it was emphasized that ease-of-use and the consistency of the toolchain are the two major points to be tackled during further developments in the project. Therefore, activities during the last reporting period were aimed in that direction.

Besides the cell design in simulation, which includes the visual tools to manipulate poses and objects in general, the transition to the real cell is described. This important step towards a consistent toolchain, allows one continuous sequence of work steps, from the initial cell design in the simulation system to the execution in the real ReconCell. A video is attached to the deliverable that demonstrates the measures described in the sections below. It can be downloaded from

`https://abr-svn.ijs.si/ReconCell/Deliverables/D4.3/D4.3.mp4`

2 Visual editing of equipment and cell configuration

Existing approaches of visual editing tools [3] in the VEROSIM® software were adapted and enhanced to meet the requirements of the ReconCell project. The goal of this action was the simplification of operating the simulation system during programming tasks (i.e. ease-of-use).

Since cell equipment and the single parts for specific use cases need to be added to the simulation model individually, the ReconCell Model Library, including the base model data imported from CAD files, was enhanced. To support the visual editing, each model was enhanced separately with semantic information about its respective properties regarding (e.g.) docking ports, collision hulls or its kinematic behavior. For further use case development the Model Library was reorganized and assembly parts and cell equipment for all use cases were added. The use of the Model Library is further demonstrated in deliverable D1.2.

Docking ports provide an interface to the user and allow to create connections between pieces of cell equipment. Two at a time can be selected and the respective parts are brought together. That way the workcell can be assembled and reconfigured for the needed requirements defined by the operator. Cell equipment available in the Model Library includes at least one docking port per part. Thus, beams can be connected, actuators can be added to the cell and assembly parts can be moved onto the trolleys. In case of the beams, multiple docking ports were defined to allow different cell layouts and an easy change process between configurations. The idea of docking ports is not new and was already successfully applied in another (non-industrial) project. However, docking ports were enhanced to meet the requirements imposed by the workcell design in ReconCell. After selecting two ports, their frames are identified such that the second is moved to and then stored below the first selected sub-model. Thus, both parts can be modified as one entity after they were connected. Since a limited number of docking ports were added to the parts in the Model Library, it might be necessary to fine-tune the position of a part. Especially the beams used for the cell construction might require the movement of attached parts along their lengths. The developed 3D gizmo allows for these fine-granular adjustments.

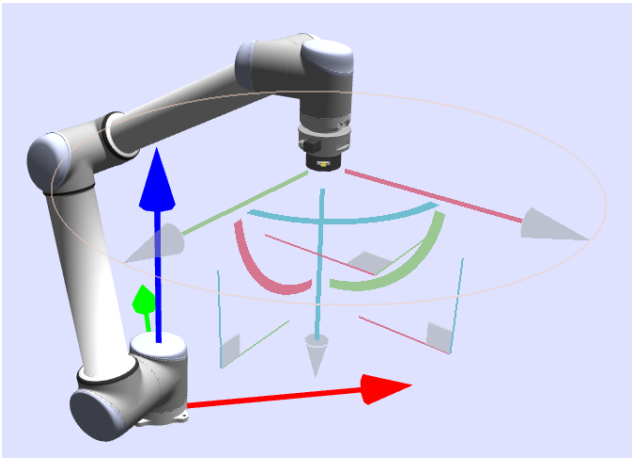

Figure 1: Robot in teach-mode with 3D gizmo at the TCP

Likewise, the video demonstrates the application of the visual guidance feature for kinematically modeled actuators in the simulation environment. Since kinesthetic guidance is a widely adopted method of teaching poses by demonstration, the same behavior was implemented in

the simulation system. For example, it is beneficial to guide the robot to a tool or workpiece while immediately getting feedback on the feasibility of the action (i.e. reachability, potential collisions).

In deliverable D4.2 the 3D gizmo was introduced to manipulate frames visually in the simulation environment. To implement robot guidance, the same mechanism was applied to the tool center point (TCP) of the currently selected kinematic chain (see Figure 1). While modifying the TCP's frame, the inverse kinematics for the corresponding kinematic chain is continuously and automatically calculated to follow the new TCP pose.

The extension of the ActionBlock concept [2] was already demonstrated in deliverable D2.1 and D4.2. While the manual creation of ActionBlock networks is useful, the task can also become tedious. Therefore, ActionBlocks were further enhanced by a feature that creates aforementioned networks automatically out of the kinesthetic guidance. Paths between the poses are assigned from a set of basic movements types. Furthermore, the created sequences of actions can be manually adapted. This is especially useful to complete complex tasks, like grasping actions. The *ActionBlockTools::PathMovementGenerator* is used to create and connect a set of ActionBlocks. A reference to a *Path* is used to identify the exact sequence of movement actions to subsequently create an ActionBlock for each path segment and parameterize it. Additionally, connections between ActionBlocks are created to signal the "Finished" state of one ActionBlock to the next one in the path sequence (see Fig. 2). Since it is still necessary to modify the auto-generated sequence, e.g. because a gripping action ActionBlock has to be added, the connection between two subsequent ActionBlocks can be splitted to insert the needed action.

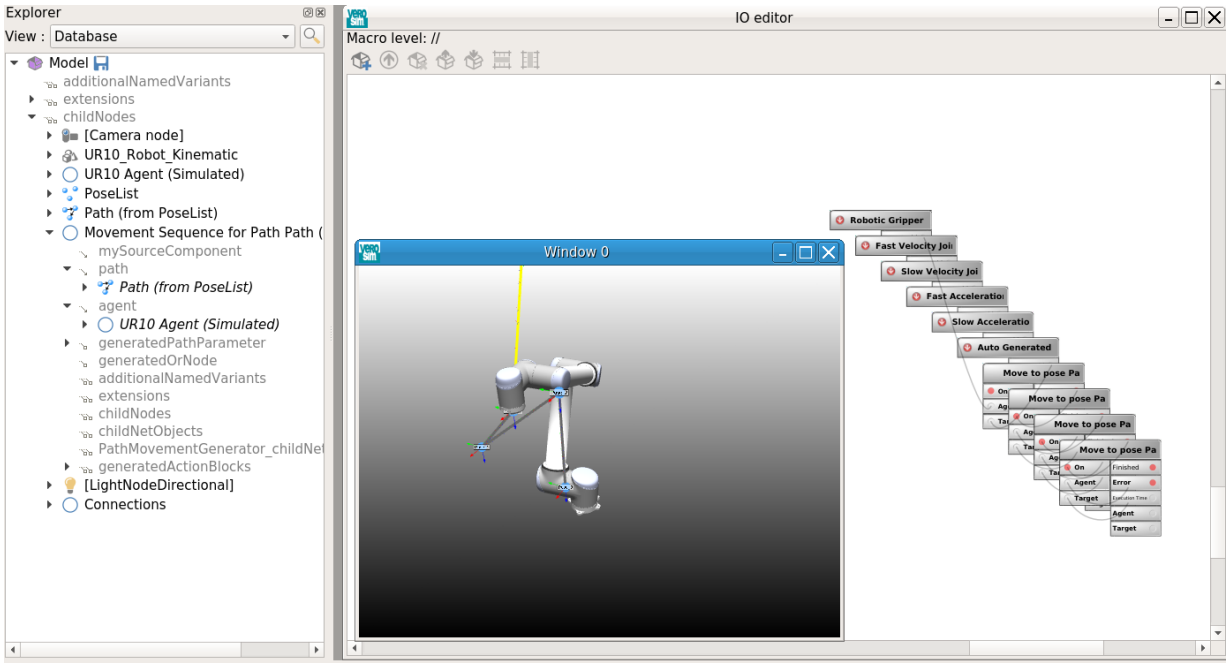

Figure 2: Automatically generated ActionBlocks given a Path and Agent

In conclusion, the visual features enable the operator to create initial cell designs intuitively and to reconfigure existing cells efficiently.

3 Transition from the simulated environment to the real cell

The transition from simulated environment to the real cell is an important step to close the open gap in the ReconCell toolchain. This was also emphasized during the review meeting.

While two separate collections of tools existed, one for the simulated environment and one for the real cell, the tasks performed during the last reporting period aimed at merging these two units to one comprehensive toolchain.

The previously mentioned semantic information is also used in the transition from the simulated to the physical environment. It is observable in the video that simulation provides useful feedback during path planning tasks, when it comes to collision detection. For two objects, the respective collision hulls are highlighted in different colors. This way, the user gets steady visual feedback during the robot motion.

The preplanned path causes a collision between the actuator and one of the cell beams, which would be fatal and costly in reality. Consequently, the operator is informed that measures must be taken, before a transition to the real cell is reasonable and safe.

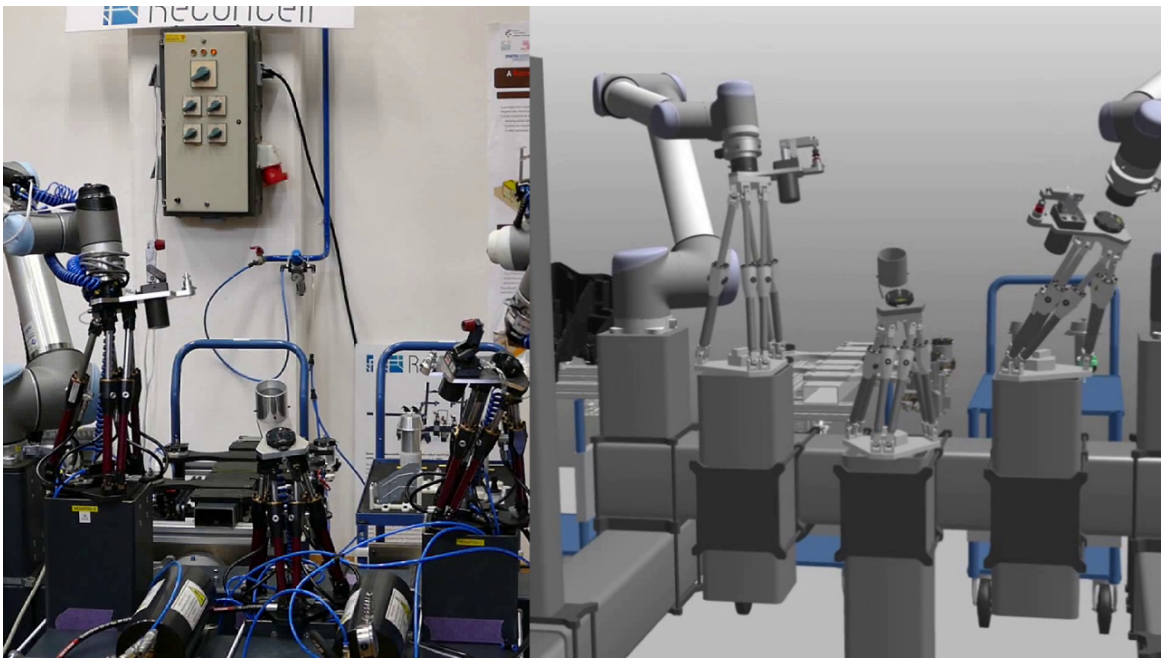

Figure 3: Side-by-side view of a ReconCell in real and simulated environments

The actual transition is implemented in two ways, both building upon ROS messages and actions.

State replication in simulation

The ReconCell software infrastructure defines a set of ROS messages providing detailed information on the current cell status. Besides the measured joint states of deployed robots, the gripper and tool exchange status is observable throughout cell operation. In deliverable D6.1 the software architecture and the ROS state publisher was described. It includes a status message (bitmask) for the Input/Output (I/O) state encoded in a ROS *std_ msgs::double* messsage. To represent the I/O state in simulation, an interface was created for the simulation system

that mirrors the bitmask as boolean outputs on the IO-Board. Likewise, the robots' states published as ROS *sensor_ msgs::Joint_ State* messages were integrated in the simulation system. It provides joint-specific positional data accessible through the IO-Board and also directly applicable to a model of a serial kinematic chain.

The simulation system is capable of accumulating all these status messages to replicate executed tasks in real-time. Spatial information of the robots' TCPs is used to identify grippers and touchpoints that need to be triggered during gripping actions. While the simulation is running, each robot's position is updated continuously and mirrors the real robot. Due to the fact that the real cell's setup mirrors the setup in simulation, the spatial information of all calibrated parts is available to identify the part an action is to be carried out on. Gripping actions are identified by the logical state change of the robot I/O and are also mirrored in simulation by triggering the simulated grippers.

Thus, analysis of cell operation becomes possible. This opens up a wide range of possibilites for testing, optimization and verification, as all robot states can be displayed during a real task. For demonstration purpose, the ELVEZ use case was used to verify the functionality of state accumulation.

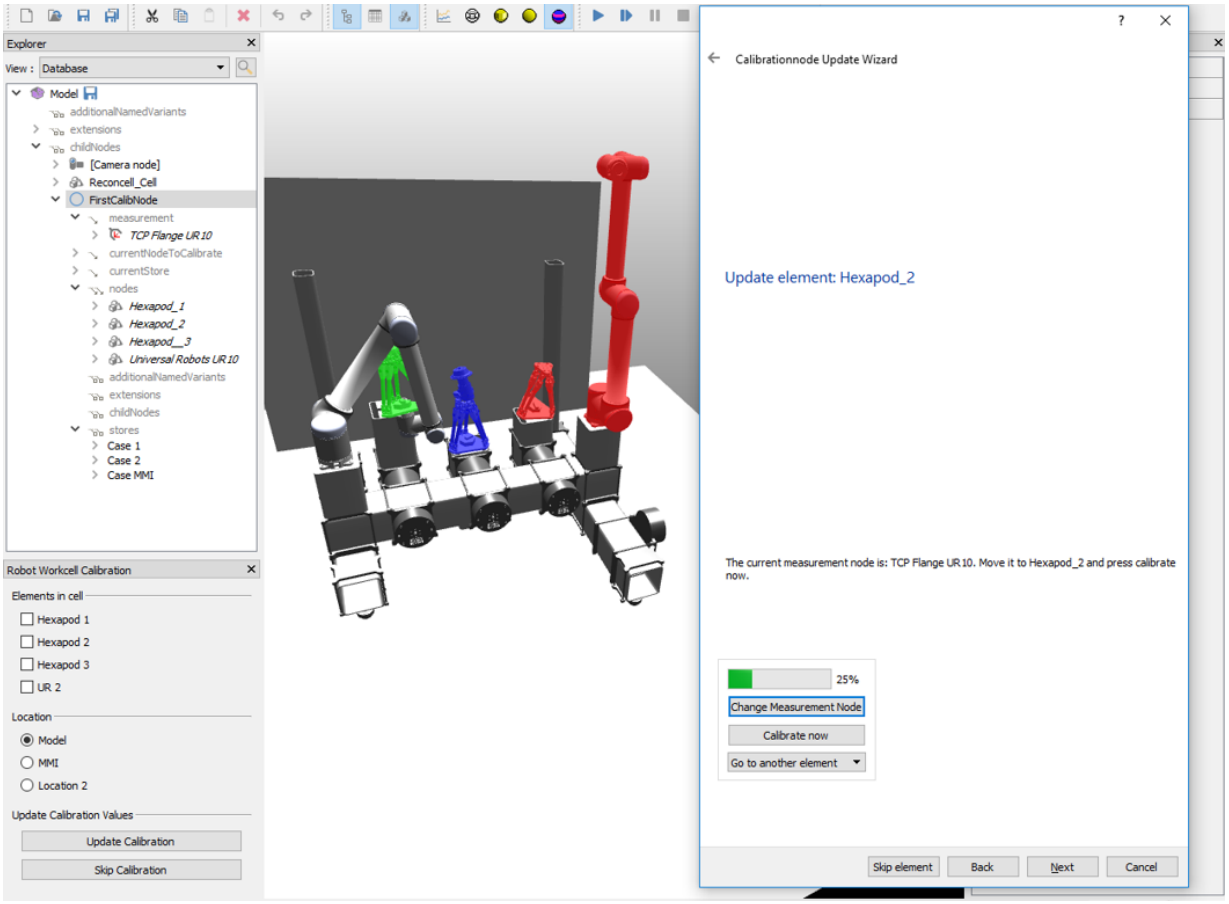

Figure 4: Workcell calibration wizard in the simulation system (green: calibrated, blue: calibrating, red: uncalibrated)

For the state replication, calibration of parts and equipment is essential. The same holds true for the control of the workcell operations, which shall be described in the following section. The poses of parts relative to a single base frame might be different or slighlty off in simulation

compared to the real cell. This might be due to inaccurate placement, which consequently needs to be compensated in the simulated environment to allow reliable operations. Therefore, a calibration node (*CalibrationNode*) and a measurement node (*DataStore*) were introduced. They reference objects in need of calibration and hold the calibration data (*CalibrationData*) respectively. During calibration, a store can be selected to place calibration data. This allows using the simulation system for the creation of cell configurations and afterwards store the calibration data for setups of the same cell at different locations. In addition, the stores include an interface to a MongoDB database to access general data available in the ReconCell infrastructure (see Deliverable D6.1) or store calibration data in a central database. The calibration data is calculated by using a measurement node, which is a calibrated tool available in simulation as well as in reality. Since the UR10 robots are calibrated during assembly and CAD data as well as Denavit-Hartenberg parameters are available to create a reliable simulation model, one of the in ReconCell deployed robots is used for measurements.

The robot is moved to the individual parts one by one and a calibration data object is stored for each. Afterwards, a store contains data objects composed of offset frames between the modelled object and the measured pose. To simplify the workcell calibration, a user interface was implemented (see Fig. 4) to provide an easy way to calibrate cells consisting of large amounts of objects, while maintaining a clear view of calibrated and uncalibrated parts.

Simulation-based control

Besides the passive behavior of the simulation system during cell operations, services can be instructed to actually control the actuators in the cell. The ReconCell infrastructure defines a set of distinct action services in the ROS network ([1]). Corresponding clients are embedded in the robots working in the simulated environment and call the actions on their twins in the real cell. The ActionBlock concept introduces agents providing behaviors that can be triggered by actions (see Fig. 5).

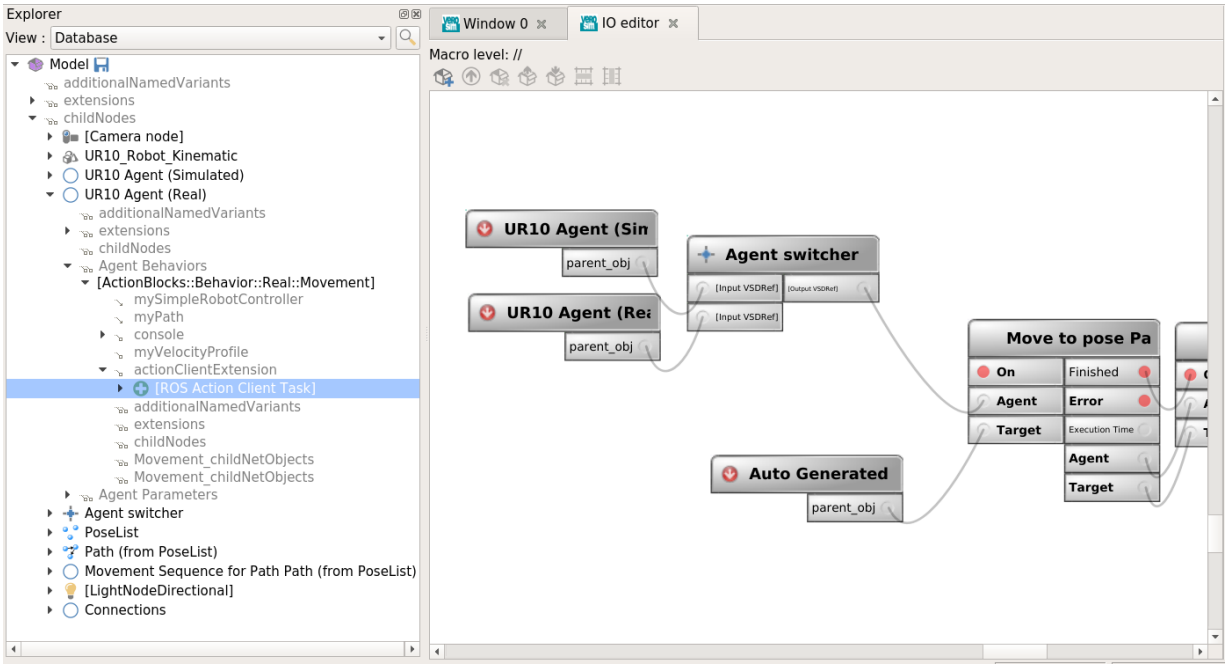

Figure 5: Simulated and Real agent

Since these behaviors are already successfully applied in controlling the operation of a simulated workcell, behaviors were extended to also apply to the instrumentation of actuators in the real workcell. Therefore, two cases were implemented:

- Movement behavior for serial kinematic chains
- Gripping behavior for grippers and tool-exchange systems

In simulation, an agent providing a simulated behavior represents an UR10 robot in a ReconCell. This behavior is triggered as soon as the current action in an ActionBlock network, that is responsible for a robot movement, transitions its internal state to execution [2]. The behavior then calls a robot controller that issues the path planning mechanism and interpolates robot positions to move the actuator to a target position. Similarly, the additional behavior for the real robot movement instructs the ReconCell infrastructure to carry out certain motion commands. The real movement behavior is implemented by utilizing ROS Action Clients. Services for specific movement types are offered by Action Servers integrated in the ReconCell infrastructure. Therefore, the new ActionBlock behavior instantiates a ROS Action Client that is connected to the ReconCell Action Server and calls a service for either Continuous Path (CP) or Point-to-Point (PTP) robot motion. For a successful call, ROS Action Services require a client to provide a goal specification that has to be reached. If successful, the ActionBlock state is set to "Finished" and execution of the network is continued. Otherwise an error state is reported and execution stops. The goal for PTP movements contains the joint-specific target position, while the goal for CP movements contains the target pose in task-space. For both goal types, a percentage value is set to specify the velocity override to impose a speed limit for the controlled robot. Thus, robot motion is transitioned from simulation to reality by connecting either the simulated or the real agent in the ActionBlock program (see agent switcher in Fig. 5).

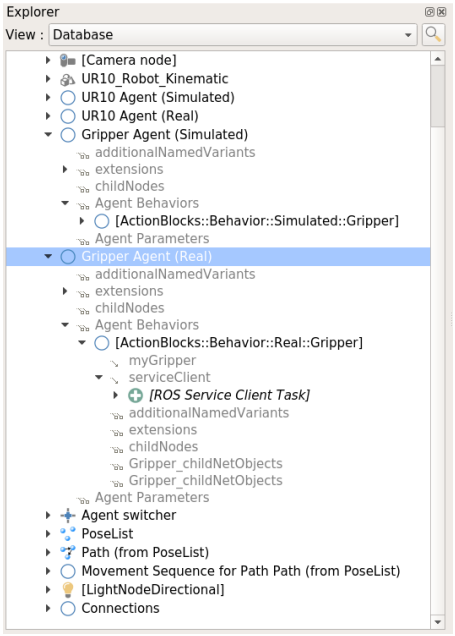

Figure 6: Simulated and Real Gripper agent

Likewise lock and release actions for the tool exchange systems and grippers can be issued. Each gripper has its own agent (see Fig. 6) defining what behavior must be called to complete a gripping action (e.g. lock, release) represented by a logical boolean state. In simulation, ActionBlocks for gripping actions trigger simulated grippers to attach either a touchpoint within limited reach, or a touchpoint that is specifically defined. To realize the transfer from simulaton to reality, a real gripping behavior was created to set the gripper specific Input/Output (I/O) state that signals the hardware to lock or release the corresponding mechanism.

Besides apparent and widely used finger grippers, the tool exchange systems and simple fitting tools are considered as grippers, while the latter incorporates no actively controllable mechanisms, but rather braces a workpiece by creating a sufficient level of friction during robot motion. The ReconCell infrastructure implements ROS Services including an I/O Service that can be used to set a robot's I/O state. The ActionBlock behavior for gripping actions therefore instantiates a ROS Service Client and issues a configurable bitmask to specify the output state triggering the hardware grippers.

A cell operator using the visual tools to program action sequences can create a cell configuration and implement assembly tasks in the simulation. Afterwards, the transition to the real cell can be performed easily.

References

[1] T. Gašpar et al. "Rapid hardware and software reconfiguration in a robotic workcell". In: *18th International Conference on Advanced Robotics (ICAR)*. Hong Kong, 2017, pp. 229–236.

[2] D. Losch and J. Roßmann. "Visual Programming and Development of Manufacturing Processes Based on Hierarchical Petri Nets". In: *3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*. 2016, pp. 154–158.

[3] C. Schlette et al. "3D simulation-based user interfaces for a highly-reconfigurable industrial assembly cell". In: *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016.