Open-Digital-Industrial and Networking pilot lines using modular components for scalable production

Grant Agreement No	**:**	**101017141**
Project Acronym	**:**	**ODIN**
Project Start Date	**:**	**1st January, 2021**
Consortium	**:**	**UNIVERSITY OF PATRAS – LABORATORY FOR MANUFACTURING SYSTEMS AND AUTOMATION** **FUNDACION TECNALIA RESEARCH & INNOVATION** **KUNGSLIGA TEKNISKA HOEGSKOLAN** **TAMPEREEN KORKEAKOULUSAATIO SR** **COMAU SPA** **PILZ INDUSTRIEELEKTRONIK S. L.** **ROBOCEPTION GMBH** **VISUAL COMPONENTS OY** **INTRASOFT INTERNATIONAL SA** **GRUPO S21SEC GESTIÓN, S.A.** **FUNDACION AIC AUTOMOTIVE INTELLIGENCE CENTER FUNDAZIOA** **DGH ROBOTICA, AUTOMATIZACION Y MANTENIMIENTO INDUSTRIAL SA** **PSA AUTOMOBILES S.A.** **AEROTECNIC COMPOSITES SL. U.** **WHIRLPOOL EMEA SPA** **WHIRLPOOL MANAGEMENT EMEA SRL**

Title	**:**	ODIN Digital Component virtual commissioning framework – Initial Version
Reference	**:**	D3.2
Availability	**:**	Public
Date	**:**	27/01/2023
Author/s	**:**	VIS, TECNALIA, LMS

Summary:

The purpose of this document is to present the first prototype of the integrated digital tools for virtual commissioning and control.

Table of Contents

1. LIST OF FIGURES

2. LIST OF TABLES

3. EXECUTIVE SUMMARY

The target of this document is the presentation of the ongoing work in the task 3.5 of the ODIN project. This task targets the virtual control and commissioning of the pilots. The tasks extend the work developed within WP3 to the development of interfaces to validate pilot requirements using virtual control and commissioning.

At the current state of the project (M24) the tasks are ongoing in two interrelated lines. Virtual commission is developed in one line to address the validation of the automation systems supported by the virtual control with the development of the interfaces and connectivity. This deliverable presents both lines starting with the introduction of virtual commissioning, its advantages and how it is implemented.

The deliverable continues with the development of the communication interfaces that are being developed on top of the simulation platform used in ODIN, Visual Components 4.0 (VC 4.0) release 4.6, and the development of communication interfaces for robots UR and KUKA used in the project pilots.

The deliverable ends with the workflow introduced for virtual commissioning and the requirements regarding access to virtual robot controllers (VRC).

4. INTRODUCTION

Virtual Commissioning is the phase before the commissioning of a manufacturing system. As presented in section 5, it is an important phase of the automation system lifecycle, and the availability of technologies that enables will enhance productivity, avoiding costly mistakes, accelerating production ramp-up and achieving quality.

Aligned with the objectives of ODIN project, the further development of technologies and methodologies that support virtual commissioning and control and the introduction of them in the pilots will enhance productivity, accelerating the deployment of automation and robotics technologies.

This deliverable focuses on the ongoing work in task 3.5 until month 24 towards the virtual control and commissioning of the pilots. In this document the completed work is presented by starting with a brief overview of virtual commissioning and the relevance under the ODIN project. The advantages regarding productivity, use of resources and quality, avoiding errors presented in section 5.1, supports the development and deployment of virtual commissioning technologies.

The deliverable continues with the development of interfaces for communication developed in Visual Components 4.0 (VC 4.0), release 4.6, for enabling the virtual control and commissioning. The development of the control has been done using the open interfaces provided by VC 4.0, in this case the ROS 2 connectivity (section 5.2.1), and the communication interface for the UR and KUKA plugin for virtual commissioning.

The deliverable ends with the robot virtual commissioning in section 6, which includes the workflow for robot virtual commissioning.

5. VIRTUAL COMMISSIONING

To have a common understanding about virtual commissioning, it is required to define what commissioning means in discrete manufacturing automation. According to [1], commissioning is defined as "the task to put the mounted products on time in readiness for operation, to verify their readiness for operation and, if readiness for operation is not given, to establish it". Commissioning is the final part of the system development and delivery process that results in a fully operational and tested system ready to use, which can be delivered to the customer [2].

In practice, commissioning of automation systems includes various procedures to check, inspect and test every operational component of the system, from physical fit of components and connections of electrical wiring to correct operation of work cells and the system as a whole. For controls, commissioning the activities include correction of software errors, correction of addressing failures, teaching of sensor positions and adjustment of parameters such as speeds [2]. Out of the total commissioning phase, time spent on control software and electrics is by far the most time-consuming part with up to 90% share. As Figure 1 shows, the utilization of virtual commissioning reduces considerably the time improving production ramp-up, integration and avoidance of errors. [2], [3].

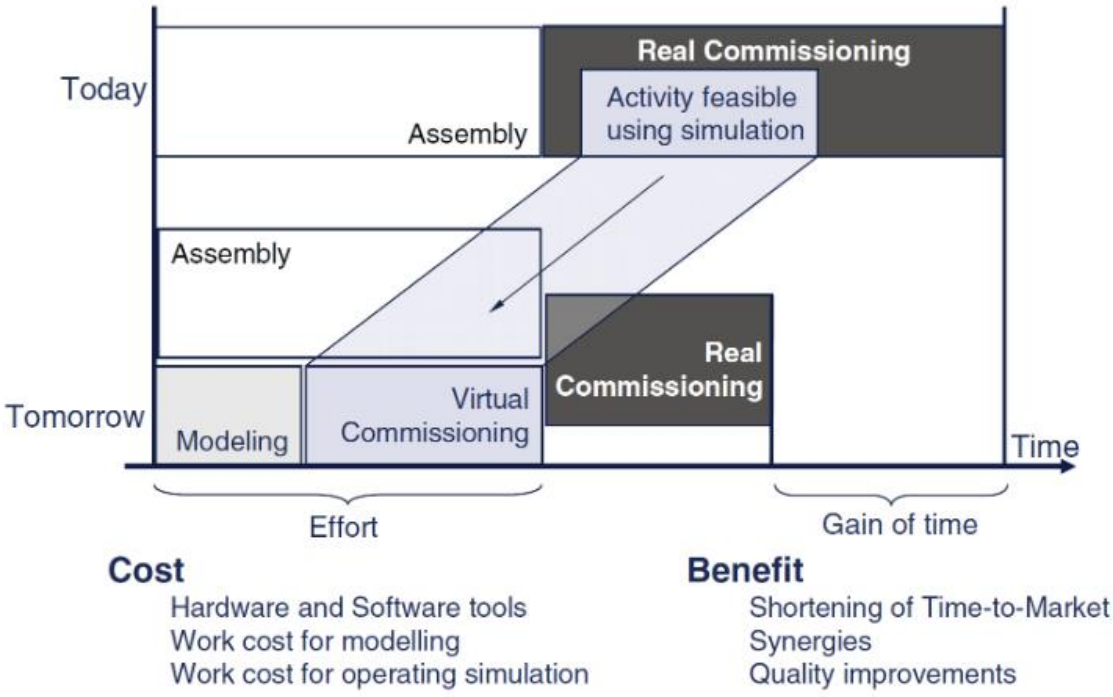


Figure 1. Engineering project with and without Virtual Commissioning [3]

The system lifecycle presented in deliverable 3.2, (Figure 2) introduces the virtual commissioning phase in the overall system lifecycle, and its integration within the simulation environment through connectivity technologies and its application in the digital twin.

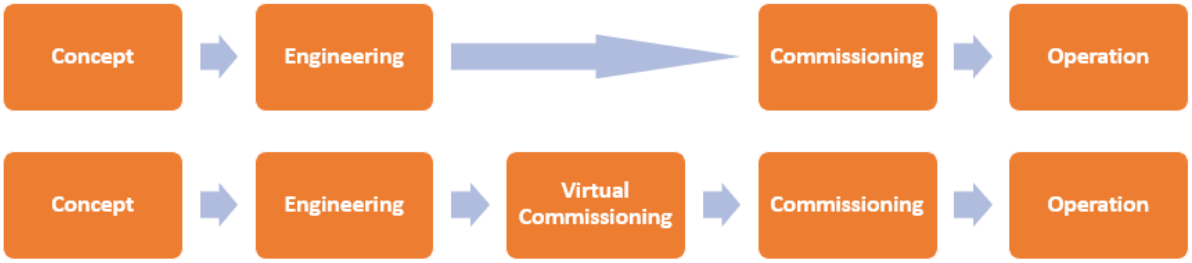


Figure 2. System lifecycle used in ODIN with and without virtual commissioning.

5.1. Virtual Commissioning in ODIN

Task 3.5 in ODIN will extend the work developed in previous tasks to virtual control and commissioning of the pilots. To achieve this objective, new communication interfaces are developed inside the simulation platform, provided by Visual Components 4.0 (VC 4.0), which allows communicating the DC with the OC to enable virtual control of the pilot in the simulation environment.

Previously to the development of the interface, the work has been focused on the identification of the interfaces, and data models used in other tasks, and extend it with the pilot requirements of work package 5. As mentioned in the description of work, the interfaces developed will be validated in the three pilots to ensure its validity during the project.

The development of the virtual commissioning in ODIN is considering the pilot requirements, in addition to consider the interoperability requirements with the rest of the modules developed in the project.

The three pilots in ODIN contain different automation equipment which can be validated during the virtual commissioning phase:

- Robot control
- PLC
- Sensors
- Cameras
- etc.

Visual Components 4.0 Premium, particularly the release 4.6 deployed in M24 in ODIN, provide the communication feature, described in more detail in section 5.3, which allows the development of communication plugins. Currently OPC UA is available and allows the validation and virtual commissioning of systems that supports that communication protocol.

Two plugins have been further developed for supporting the task in the ODIN project in the virtual commissioning of the robots, the UR plugin (5.3.1) and the KUKA plugin (5.3.2).

In addition to the communication interface, Visual Components 4.0 provides two open interfaces (5.2), .Net and Python. The open interfaces are the based to develop the plugin for ROS2 connectivity (5.2.1) that in addition to connect with ROS 2 is the base for the communication through OpenFlow.

Summarizing, Figure 3 presents the interfaces in use in ODIN until M24. While the virtual control and validation has been focused on the development of ROS2 through the .Net interface, the virtual commissioning of the robots (UR and KUKA) have been achieved using dedicated plugins to support the virtual commissioning in the white goods and aeronautics pilots.

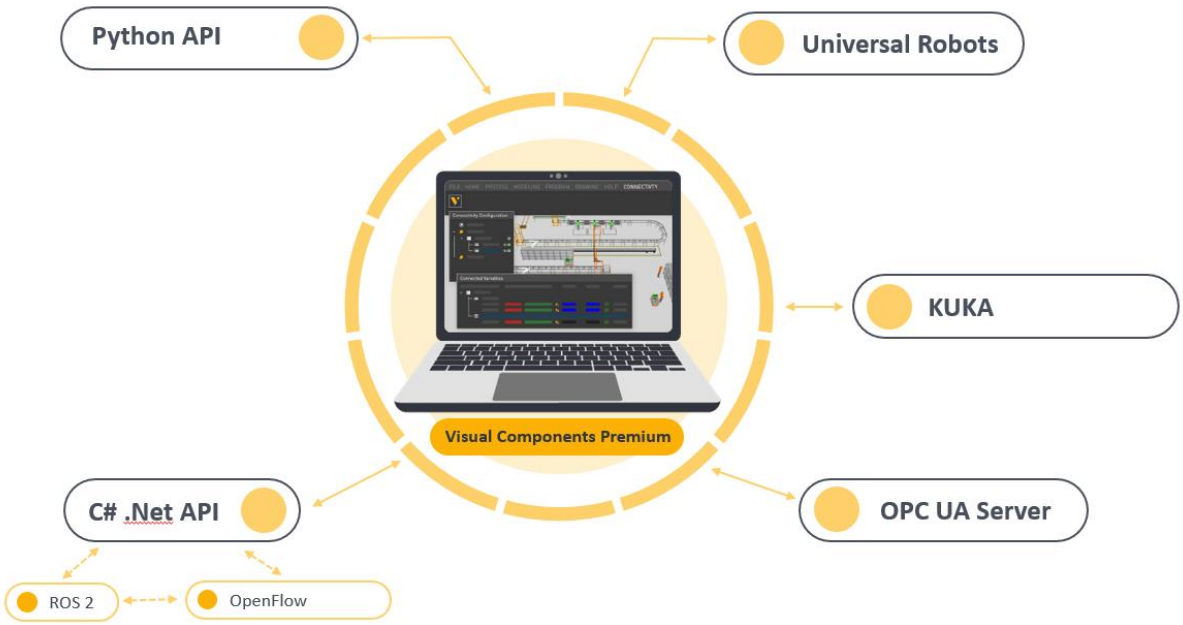


Figure 3. Overview of the communication interfaces used in ODIN for Virtual Commissioning

5.2. Open interface

VC 4.0 provides two open interfaces, .Net and python (Figure 4), which allow the development of the required interfaces. Both interfaces provide an extensive documentation for developers available through VC 4.0 user interface.

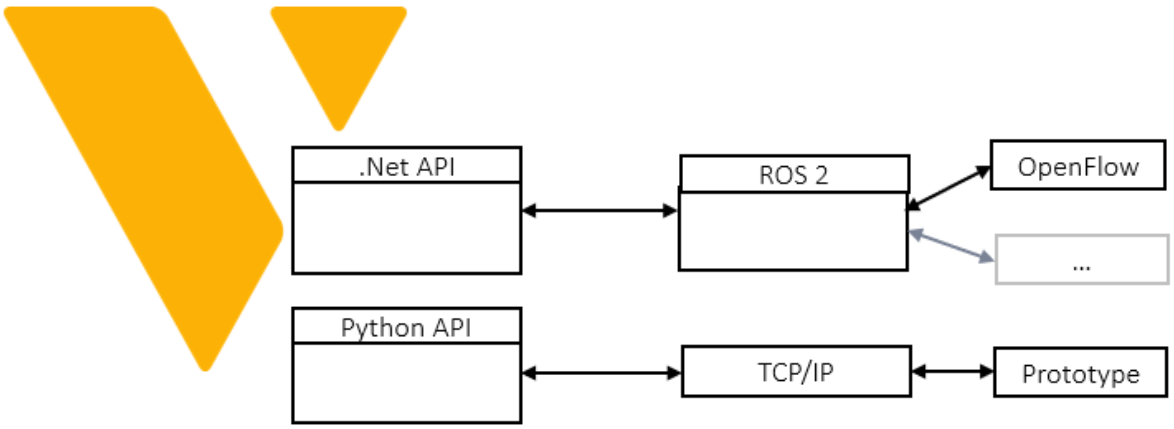


Figure 4. Overview of the open interfaces and current development within ODIN

While the python interface allows to create situation components and the development of add-ons, the .Net interface is targeting the development of plugins and extensions.

For the purposes of the ODIN project, the python interface is targeting the development of simulation models and prototyping initial communication interfaces through TCP/IP or dedicated interfaces. .Net interface has been used for the development of ROS 2 connectivity and robotics communication interface.

5.2.1. ROS 2 Connectivity

Using the .Net interface, a plugin has been developed. This plug-in that will be further developed until the end of the task, following the ODIN requirements, supports, publishes, and subscribes to ROS2 topics as showed in Figure 5.

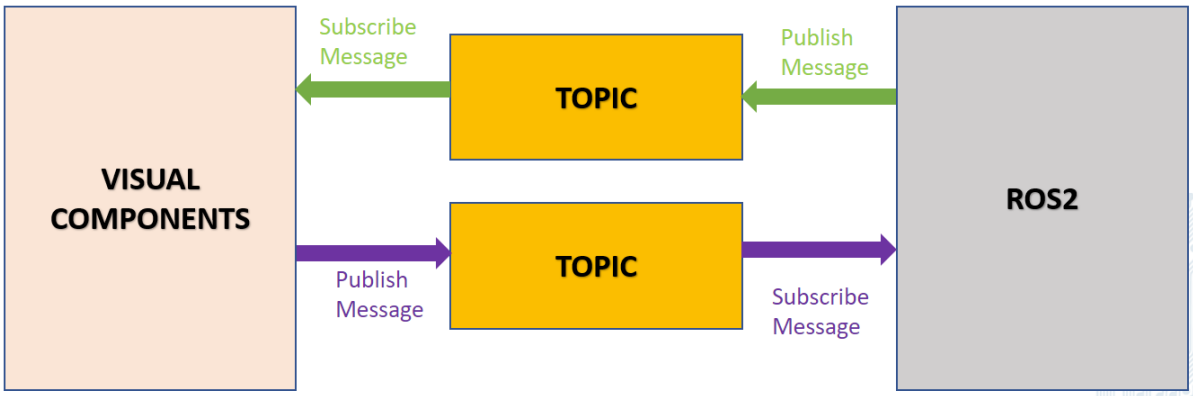


Figure 5. Connectivity schema between Visual Components and ROS2

The communication through ROS2 also allows the connectivity between VC 4.0 and OpenFlow, which also supports ROS2 interface. In that way it is integrated though the ODIN network component provided by OpenFlow with the rest of the ODIN connected components.

An example of the ongoing implementation is the use case showed in Figure 6. In this use case, joint goals are received from ROS2 to VC 4.0. The message is received through the topic and VC 4.0 creates trajectory points and statements. Once these are created, within VC 4.0 is checked reachability, joint configuration and singularity. After this, VC 4.0 sends the interpolation time through the topic, which is received by ROS2, request id and success message (Figure 6).

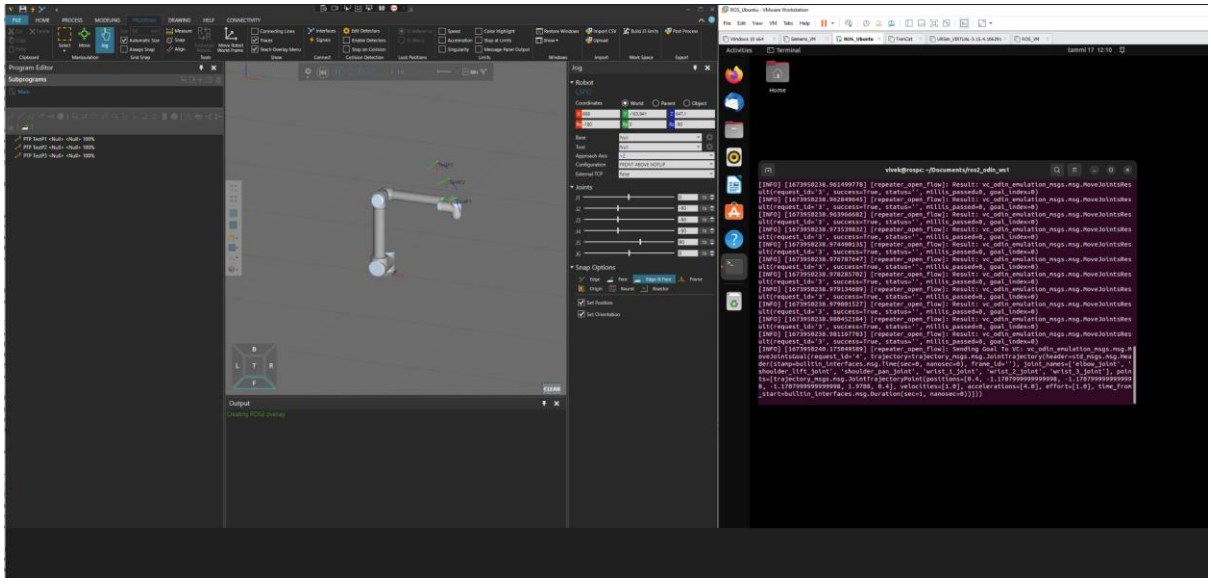


Figure 6. Workflow of ROS2 message to VIS sending joint goal to obtain interpolation time

5.3. Communication interface

The communication interface is a feature provided by VC 4.0 that allows the development and deployment of connectivity plug-ins. This feature and its commands can be accessed from the Connectivity tab accessible through the user interface of VC 4.0.

The feature provides the commands described in Table 1, and allows communication between the virtual systems (sensors, actuators, conveyors, robots, machines…) in the virtual environment and the controllers which can be real or virtual.

Table 1. Commands provided by the communication feature of Visual Components 4.0

Commands	Description
Add Group	Adds and lists a new variable group with a selected connection.
Add Server	Adds a new connection for a selected plugin.
Add Variables	Opens an editor for connecting simulation variables to server variables.
Clear	Removes all connections for each plugin.
Disconnect	Disconnects Visual Components Premium 4.6 from a selected connection.
Edit Connection	Displays options in a task pane for editing or troubleshooting a selected connection.
Export	Exports the configuration of all connections in an XML format.
Import	Imports an XML or CFG file that defines the configuration of one or more connections.
Reconnect	Attempts to reconnect Visual Components Premium 4.6 to a selected connection.
(Server) Remove	Removes a selected connection.
(Variable) Remove	Removes a selected variable group.
Restore Windows	Restores the workspace of the current view to its default setting.
Show	Displays a list of panels that can be shown/hidden from the current view of the workspace.
Show Variables	Shows a panel for managing the connection between simulation and server variables.

The current work towards the development of communication interfaces for ODIN are targeting the connectivity with UR (5.3.1) and KUKA (5.3.2) to match the robot requirements towards virtual commissioning in the white goods pilot and in the aeronautics pilot.

5.3.1. UR plugin

Support for UR connectivity has been developed within ODIN, developing a plugin which allows connecting the Universal Robot controller through RTDE interface [1](Real Time Data Exchange). The plugin for the communication interface enables seamless communication between the virtual robot and the virtual robot controller (VRC).

The Real-time data exchange (RTDE) interface provides a cyclic stream of value updates from the controller and listens for inputs. The interface updates (sends and handles data packages) at a fixed frequency and is based on a binary application-level protocol transmitted over (insecure) TCP/IP socket communication. The robot controller uses TCP port 30004 for the interface. The connection

[1] More information about the RTDE interface can be found at https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/

plugin's RTDE client implementation developed uses an automatically assigned port (either by .NET or Windows) for the socket.

The basic operation principle involves two modes, configuration and run. First, the client configures with the server the data it wants to receive and data it wants to send. This is done in configuration mode. After configuration has been set, the client can request the controller to enter run mode where the controller sends the requested data at the fixed 125 Hz frequency and the client can send its data at a preferred rate. Run mode can also be paused by request of the client to return to configuration mode.

The data packages the client and controller send to one another are defined with input and output recipes in configuration mode:

- Input is data flow from client to controller.
- Output is data flow from controller to client.

The recipes contain one or more variables from a known fixed set, and the associated data packages contain values for all variables in the recipe.

The current version of the RTDE protocol supports only a single output recipe per client, but up to 255 input recipes can be defined per client. Furthermore, it is not possible to remove an input recipe without disconnecting and creating an entirely new one. This means that adding/removing a variable pair or activating/deactivating a variable group always causes a recipe update. Since the recipe update can only be done in configuration mode, the RTDE client implementation automatically requests pausing from the controller, and then either a) redefines the output recipe shared between all variable groups or b) registers a new input recipe for the activated variable group.

The RTDE protocol does not provide any way to poll the controller for value updates. This causes some limitations that differentiate the RTDE connection plugin from others. The RTDE connection plugin manages a local cache of the variable values for all configured recipes (active variable groups). This allows using cyclic update mode to read output recipe values at any desired frequency and sending whole input recipe data to the controller in event-based update mode. However, since the output recipe updates are received and input recipe data is sent asynchronously, the update delay timing functionality of Connectivity core does not really work with the RTDE plugin. The times measured are only processing times to get data in or out of the cache. That is, they do not include the network delay or even how old the received output recipe data is when the cache is read using cyclic update mode.

The plugin has been extended with the development of the post-processor which converts the robot program statements from VC 4.0 to the UR language (.urp). To validate the program Universal Robots provides, free of charge, the URSim robot controller simulator[2] which allows to load the robot program and validate to later visualize for virtual commissioning purposes as explained in detail in section 6.1.1 of this deliverable.

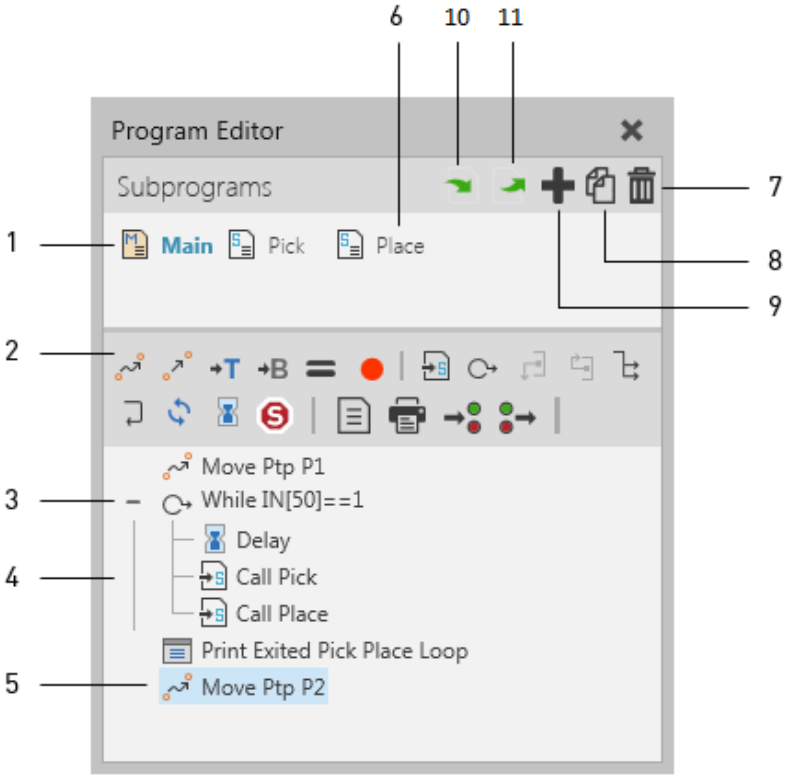
5.3.2. KUKA plugin

The Kuka plugin has been further developed to targeting connectivity to KUKA RCS (Robot Control System). The interface is deployed into the communication interface and requires a valid license for the KUKA VRC.

[2] Available for downloading from https://www.universal-robots.com/download/software-e-series/simulator-non-linux/offline-simulator-e-series-ur-sim-for-non-linux-594/

6. ROBOT VIRTUAL COMMISSIONING

The virtual environment provided within VC 4.0 provides the functionalities to program a robot and simulate the robot. The robot program editor (Figure 7) deploys the functionalities to create a robot program or modify existing ones or generated with path planning and other tools.



1. Main Routine
2. Statements Toolbar
3. Conditional Statement
4. Nested Statement
5. Motion Type Statement
6. Subroutine
7. Delete Selected Routine
8. Duplicate Selected Routine
9. Add Subroutine
10. Import Program
11. Export Program

Figure 7. Robot program editor available in VC 4.0

Once the robot program has been created and the initial simulations have been created, these robot program routines must be transferred to a language that the robot can understand. Every robot manufacturer has its own robot program language that runs in its proprietary robot controller. This is when it is required to post-process that robot program created during the simulation stage into the language that the robot can run into its controller. Within ODIN, the work done until M24 has been focused on the post-processor for UR and KUKA aligned with the good white pilot and aeronautics pilot.

After post-processing the robot program can be uploaded in the robot controller and start testing the program, but as mentioned in section 5 and aligned with the target of the task 3.5, the use of virtual commissioning will allow to validate the code and identify possible errors. To achieve the virtual commissioning, the workflow presented in the next section has been followed.

6.1. Workflow for virtual commissioning

Once the pilot layout has been created in the virtual environment and the initial simulations have been completed, it is possible to start the virtual commissioning. For achieving virtual commissioning, a workflow of seven steps presented in the next section has been deployed.

6.1.1. Workflow

1. The robot is programmed in VC 4.0 using the robot programming features (Figure 8). Modification to the robot programming to change parameters and perform new simulations can be done with the robot program editor (Figure 7).

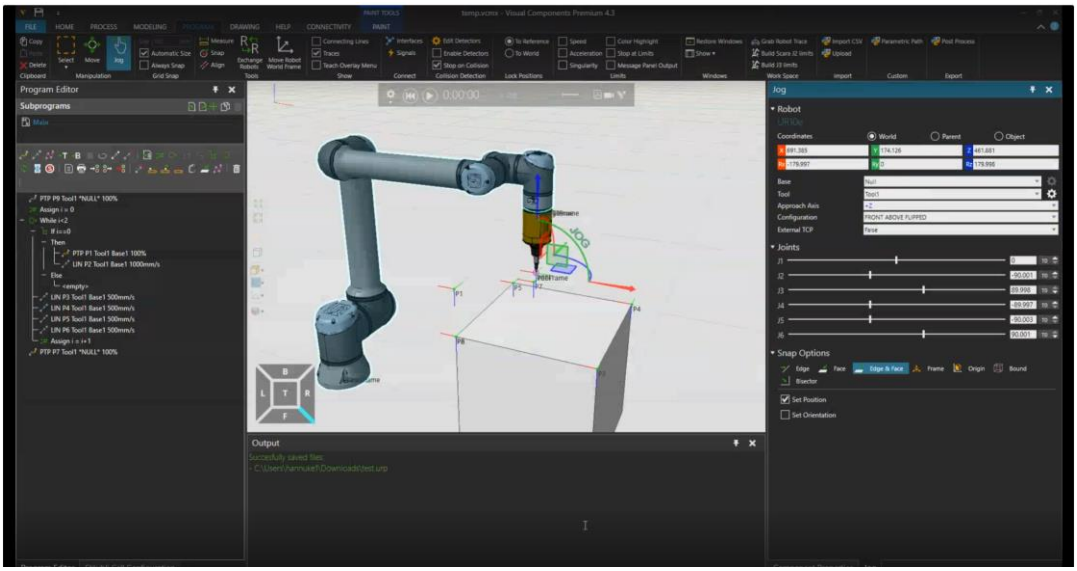


Figure 8. Robot programming within VC 4.0

2. Once the simulation results are matching the expectations, the robot program in the simulation is postprocessed to the robot language, in this case showed in Figure 9 to the UR language (.urp)

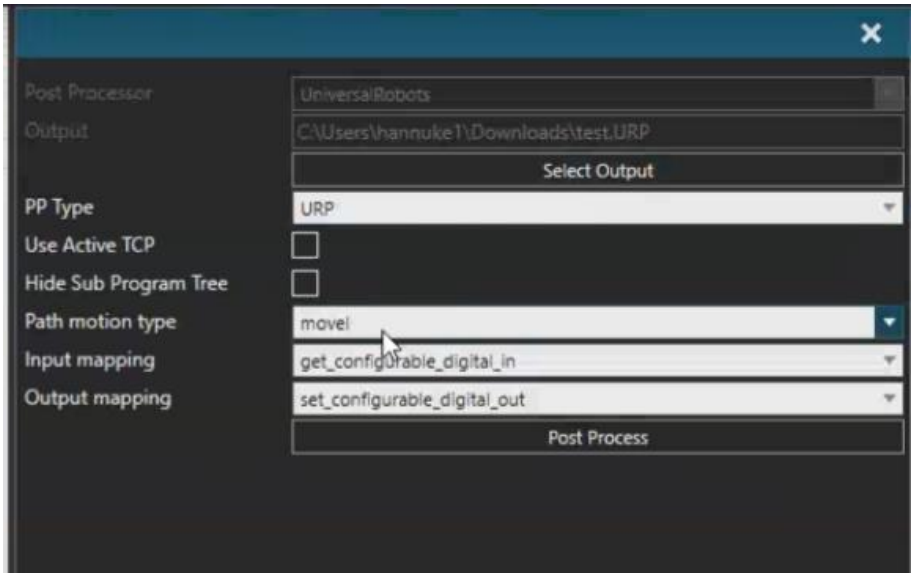


Figure 9. Screenshot of the post-process tab available at VC 4.0

3. The file with the UR program is uploaded to the UR VRC controller as showed in Figure 10.

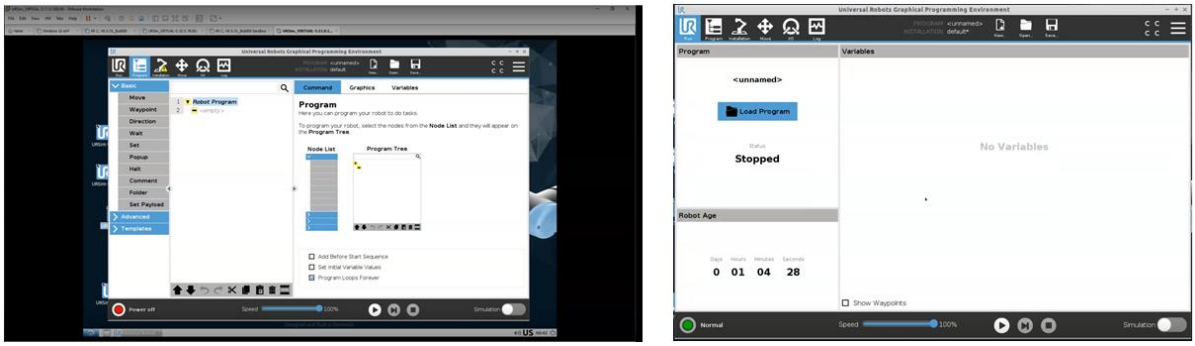


Figure 10. Upload process of the robot program into the virtual robot controller

4. The Virtual Robot Controller should be configured with the same operational parameters than the robot in the simulation (Figure 11).

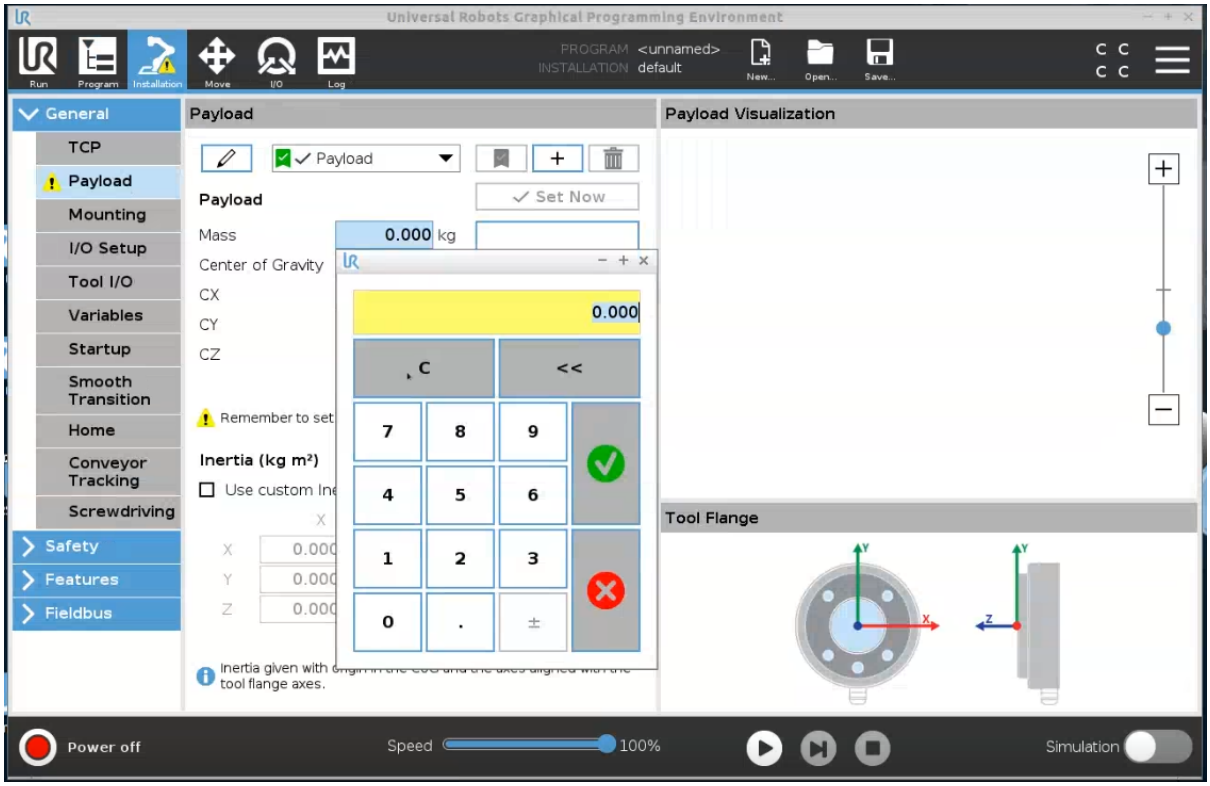


Figure 11. Configuration of the virtual controller with the same operation parameter than the simulation

5. Both robots, the one in the virtual world and the one in the virtual controller, should be in the same initial position before starting the virtual commissioning process (Figure 12).

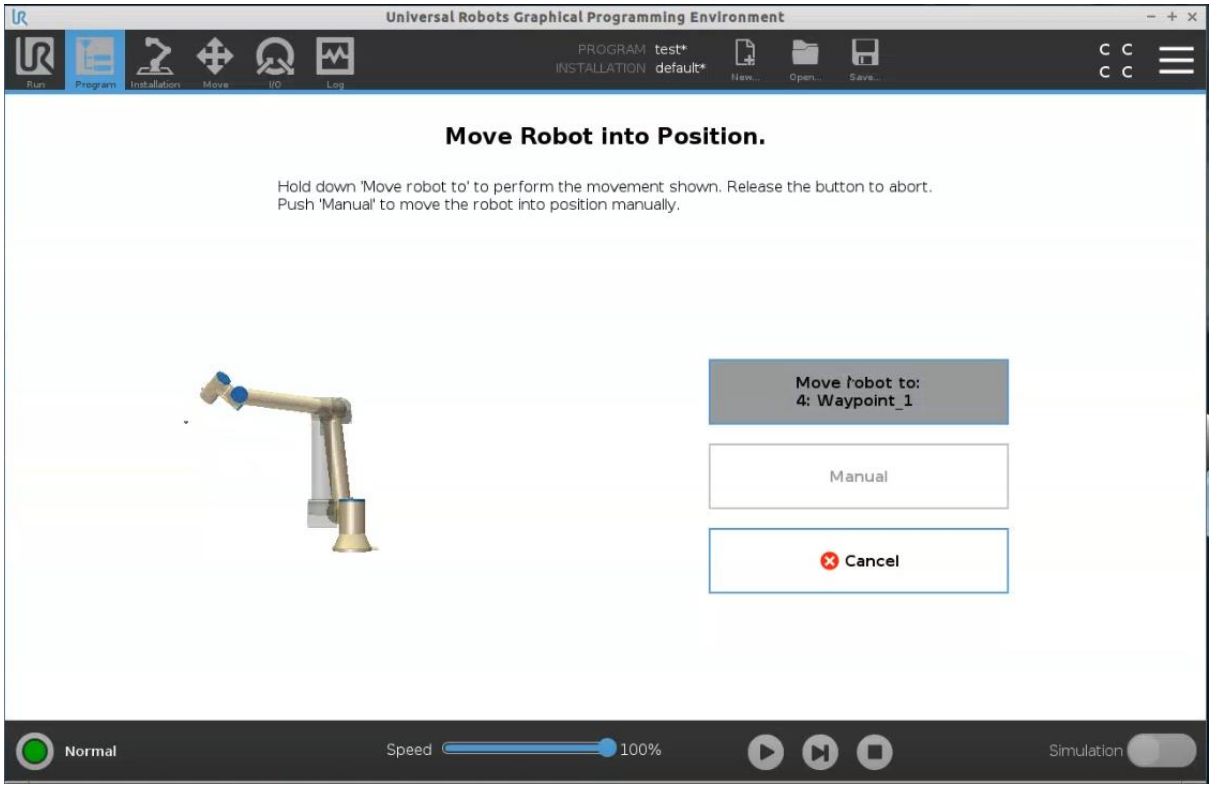


Figure 12. Position synchronization of the robots in VC 4.0

6. Before starting the virtual commissioning, the user can validate the robot program post-processed into the virtual controller (Figure 13).

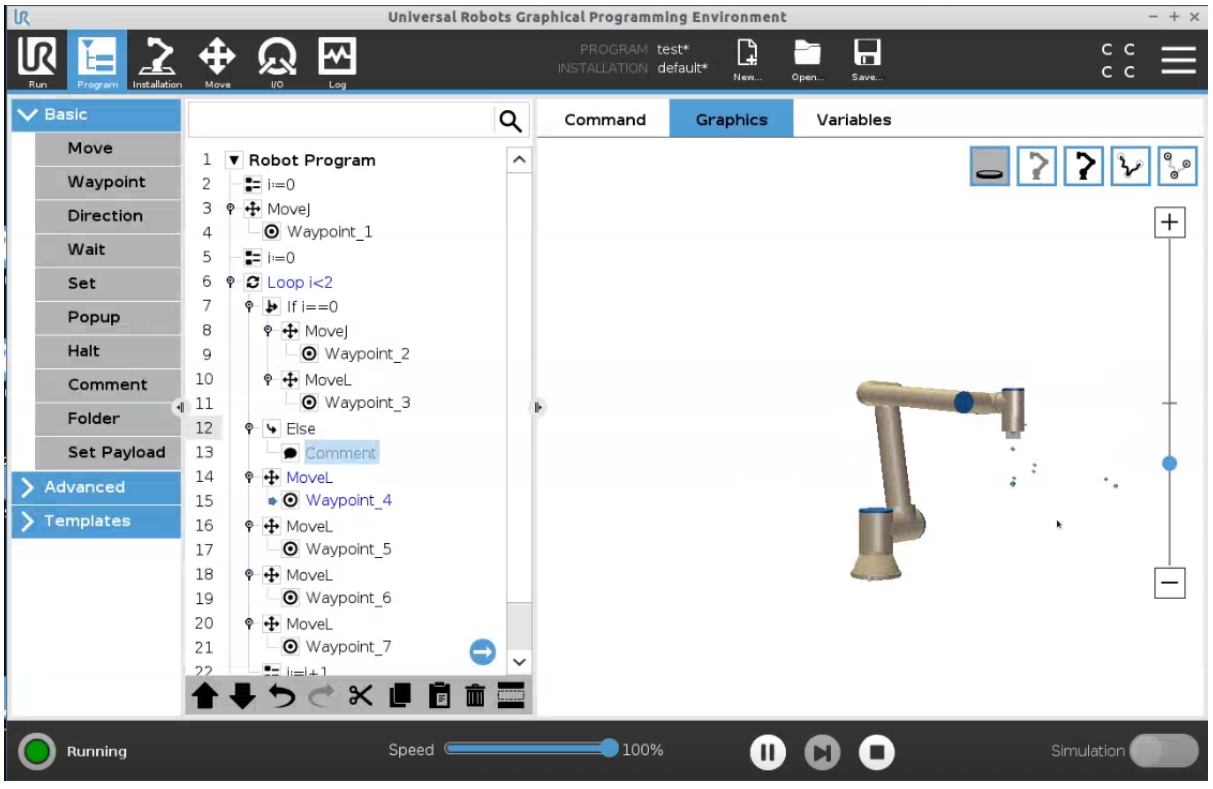


Figure 13. Validation of the robot controller

7. Finally, the virtual commissioning of the system can be done by connecting the virtual controller with the virtual robot within VC 4.0 and verify the robot program is performing as expected during the simulation phase (Figure 14)

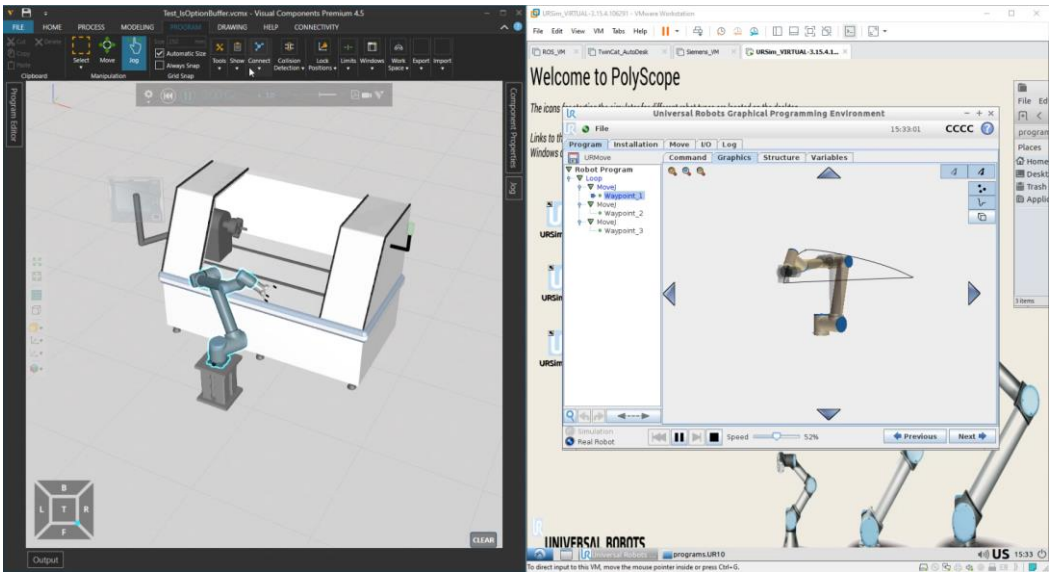


Figure 14. Virtual robot in VC 4.0 connected to the virtual controller

6.1.2. Robot requirements

As mentioned at the beginning of this section, each robot controller has its own programming language, that's why the work in ODIN has been focused on the development and maintenance of the program post-processor as well as the communication plugins.

The connectivity to the virtual robot controller is done through the communication interface, as introduced in section 5.3. Despite the connection to UR doesn't require extra configuration in the case of the KUKA robots still requires selecting the KRL (KUKA Robot Language) version in the component properties, depending on the version that the robot controller is using (**Figure 15**).

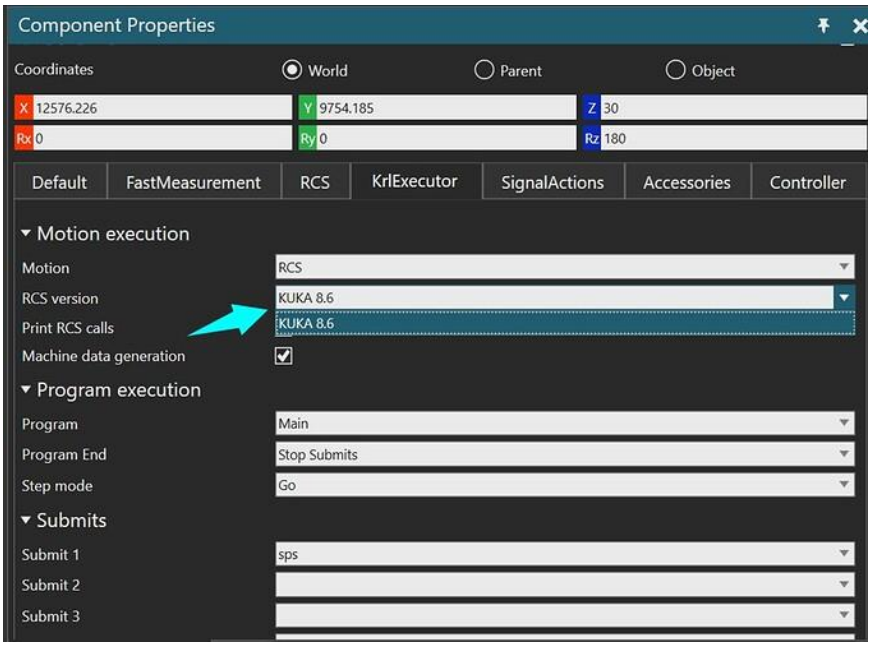


Figure 15. Screenshot of the configuration window

7. CONCLUSIONS

The work developed in task 3.5 towards virtual commissioning until month 24 has been focused on identifying the initial requirements for virtual commissioning of robots, targeting the robots used in the white good pilot and aeronautics pilot. The work has been completed and the results obtained are the base to continue the work within ODIN.

The first connectivity plug-in for ROS2 has been deployed and is operational for robot movement and messages through OpenFlow.

The work in the task continues with the implementation of more functionalities for the three pilots. it is expected that during the next months, more ROS2 services will be added and the communication interface through OpenFlow with other components will be extended.

The next target regarding virtual commissioning will be to provide the final and more enhanced version which is officially due on M36.

8. GLOSSARY

IP	Internet Protocol
TCP	Transmission Control Protocol
OC	Open Component
DC	Digital Component
TRDE	Real Time Data Exchange
UR	Universal Robots
.urp	universal robot program (extension)
RCS	Robot Control System
VC 4.0	Visual Components 4.0
VRC	Virtual Robot Controller
WP	Work Package

9. REFERENCES

[1] S. Bangsow and U. Günther, 'Creating a Model for Virtual Commissioning of a Line Head Control Using Discrete Event Simulation', in *Use Cases of Discrete Event Simulation*, S. Bangsow, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 117–130. doi: 10.1007/978-3-642-28777-0_7.

[2] S. Bangsow, Ed., *Use Cases of Discrete Event Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-28777-0.

[3] Z. Liu, C. Diedrich, and N. Suchold, *Virtual Commissioning of Automated Systems*. INTECH Open Access Publisher, 2012. Accessed: Jul. 02, 2015. [Online]. Available: http://cdn.intechopen.com/pdfs/37992/InTech-Virtual_commissioning_of_automated_systems.pdf