Agent-based framework for Reconfigurable Manufacturing Systems

A. Vinulescu, D. Sarancic, M. Kolek, S. Zelazny

Department of Materials and Production, Aalborg University
Fibigerstraede 16, DK-9220 Aalborg East, Denmark
Email: szelaz13@student.aau.dk,
Web page: http://www.mechman.mp.aau.dk/

Abstract

In the times of highly volatile global markets, manufacturers find themselves having to adapt to fluctuating demands, shorter product life cycles and high customer expectations regarding delivering a customized product. This gives rise to a new era of Reconfigurable Manufacturing Systems (RMS), that offer a higher degree of flexibility and changeability, making them more suited for competitive markets. This article tackles the challenge of developing a the software architecture enabling the operation of a reconfigurable manufacturing system, capable translating the product to a set of processes needed to create it and handle their execution. The created framework is capable of supporting any type of manufacturing process, demonstrated on the case of a multi-agent robotic assembly.

Keywords: Reconfigurable manufacturing, agent-based framework, plug-and-produce, robotic assembly

1. Introduction

The effort of the manufacturing companies to sustain competitive advantage in nowadays rapidly changing markets represents a challenging task, as the manufacturing systems need to respond and quickly adapt to fluctuations in demand, product mix and frequent introduction of new product variants [1]. The capability to operate robustly at high throughput (typical for dedicated manufacturing systems - DMS) or agilely as a response to market demand but at lower throughput rate (typical for flexible manufacturing systems - FMS), is balanced by the reconfigurable manufacturing systems (RMS) for which capacity and functionality can be changed exactly when needed. While DMS and FMS present rigidity at the capacity-functionality level, RMS presents minimized constraints for capacity or functionality, due to the ability of instating rapid system configuration changeover in response to the capacity demand.

1.1 Study case - AAU Smart Production Lab

In the light of the trend described in the previous section, Aalborg University (AAU) has in 2016 purchased the AAU Smart Production Lab (Smart Lab), an automated modular assembly to be used as a technology demonstrator and research platform for Industry 4.0. Currently the system assembles unusable phones consisting of a printed circuit board (PCB), fuses, an upper and lower housing part. It is wished that the product family will be extended to new product configurations, shown in figure 1 in order to imitate a scenario for which the assembly line needs to accommodate product mix and variation.

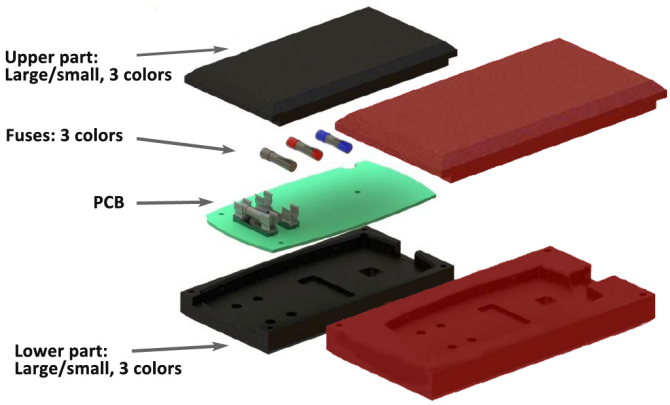

Fig. 1 The extended product

The product is assembled as it passes through as series of assembly processes performed by particular equipment (i.e. drill), referred to as process modules. The process modules are mounted on top of modular stations which have the function to transport the product by mean of conveyor belts.

To simulate eventual scenarios where the capacity throughput of the assembly line could be increased when needed, the assembly line is desired to gain abilities of fast rerouting and reorganization of resources. On a theoretical level, the capacity increase can be achieved by modularizing the different components in the robot

cell of the assembly line, shown on figure 2 in order to make them recognizable and operate independently of each other. The components which should be split into modules are: (1) the vision module which detects part orientation, (2) the PCB magazine dispenser and (3) the fuse dispenser. Once the physical reorganization of the process modules is carried out, a software architecture model becomes an indispensable part to support the physical changes. Thus, the focus of the current research is oriented towards the development of a data interconnection model.

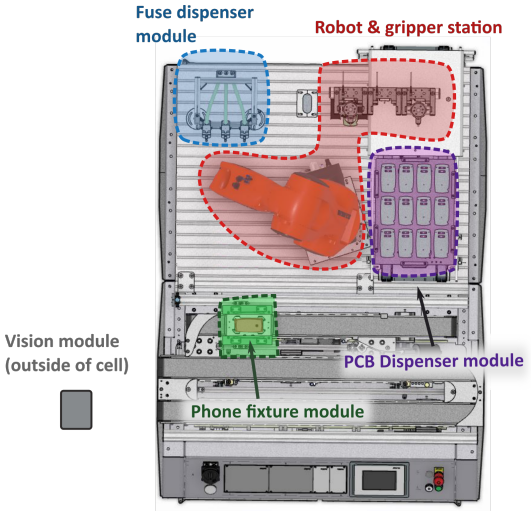

Fig. 2 Modules inside the robot cell

2. Related work

As the underlying control architecture is the true enabling factor for achieving an ideal "plug and produce" system, different strategies for meeting the growing demand for flexibility have evolved over time. Early 90's started the trend with the concept of Flexible Manufacturing Systems (FMS), [1] [2] extending the systems capabilities to handle a larger span of variety although at the expense of added cost in the system [3]. In order to avoid investing in dedicated equipment which is limited in functionality, the focus has shifted towards designing more agile systems, leading to a wave of new paradigms: Holonic Manufacturing systems, Modular Manufacturing systems, Evolvable manufacturing systems and more, [4], [5], [6], all built around the principle of moving away from centrally controlled systems towards encasing the functionality of the control system into independent modules. This allows to easier scale, change, maintain, reuse and upgrade the equipment and its software, by reducing the amount of inter-dependencies, and aiming towards more task-specific modules extending the production capability as needed.

The latest contributions to this paradigm, which are used as frame of references for this system are the EU-sponsored IDEAS (Instantly Deployable Manufacturing Systems) project [2] and it's successor the OpenMOS (Open Manufacturing Operating System) project, aiming to develop a common, openly accessible plug-and-produce system platform, independent of the specific industry [7].

While the OpenMOS project has developed a very encompassing semantic model, describing almost every thinkable aspect of operating a manufacturing system [8], there are few publicly available descriptions of how this philosophy is implemented in practice. Secondly, current literature on the robotic pick-and place operation, the modularity of which is the key changeability enabler for the Smart Lab, often overlooks the role of the cell fixture and feeder, assuming their position and type is static, and focusing in the interaction between other assets (part-robot or robot-robot) inside the robot cell.

This paper presents a software framework developed for enabling a plug & produce capability of the Smart Lab, with a special focus on the robotic pick and place process. However in order to make the solution flexible, scalable and "future-proof" the framework is developed to treat the Lab as a specific instance of a generalized theoretical model. In this way the framework is also relevant to any other assembly-line production system, especially those whose operation requires rapid and frequent changeovers.

3. Approach

In order to systematically approach the design task described in section 1 a semantic network model proposed for the Smart Lab is presented in section 4. This model creates a set of general entities used to describe a production system. Based on this formalization, a software architecture was built around it and is presented in section 5, defining all relevant components required to operate the manufacturing system. Based on this architecture a execution protocol was designed, specifying the sequence and communication during execution of a manufacturing process. This protocol, described in section 5.2 can then be extended to involve multiple agents collaborating on a task, in this case a robotic pick-and-place assembly, as described in section 5.3. Finally, having established a structure for executing an assembly and a process sequence, the concept for executing more tasks in parallel is introduced and described in section

5.4.

4. Ontological model

The field of ontology works with the formal representation of concepts, the organization of these into classes along with associated attributes and their mapped mutual relations [9] [10]. Based on similar work by Lohse et. al [8] the semantic network model of the Smart Lab created is shown in figure 3. In order to create a model that would be able to accommodate future processes and products, the production system is in its basic form modelled as two generic types of entities:

- A *product recipe*, which is a list of *processes* needed to be executed in order to assemble the product components
- Specific *skills* representing the capabilities of the process modules to execute the processes.

According to the processes family, the processes can further be classified into *simple processes* requiring only one skill (eg: a drilling station) or *complex processes* requiring the more complex cooperation of multiple skills (eg: an assembly operation.)

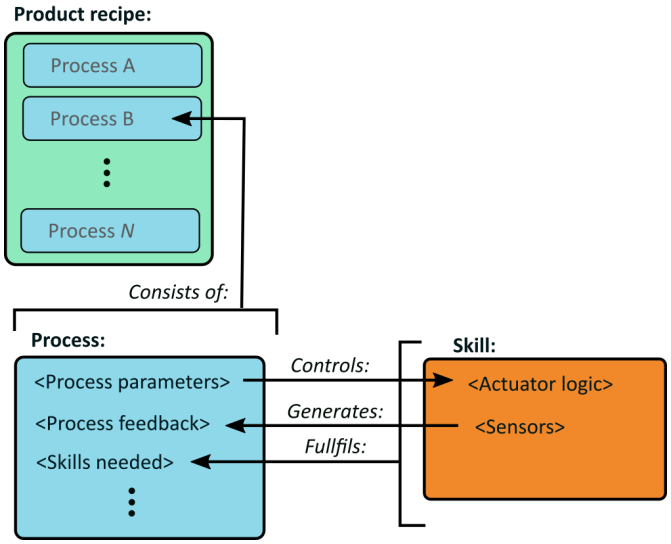

Fig. 3 Semantic model of the process and skill concept.

In a manufacturing system the skills are manifested by the process modules. With the advent of affordable industrial grade micro-controllers, it is envisioned that each process module(i.e. a machinery for drilling) performing a function will be outfitted with a microcontroller capable of presenting the process module's skills to the master control system (i.e. MES). The process module and the microcontroller would create what by Pritschow [11] would be denoted as mechatronic agent.

The mechatronic agents represent the entities which can be added or removed from an manufacturing system which enables a vast flexibility to re-configure a manufacturing system, according to the equipment and its routing needed.

The control of the different settings of the process module, (i.e. the drilling pattern and depth of the holes) is performed using specific *process parameters* passed to the skill executing them.

As depicted in figure 4, the process parameters stored in the product recipe, are passed in a string format, which allows each equipment vendor to use their preferred/legacy-dictated method to pass the parameters to their equipment by encoding them into the string, (eg. using XML or a similar markup language.)

As the actuator logic is stored on the mechatronic agent, the modularity of the factory goes beyond the mere interface compatibility of hardware, but extends to self-configuring the process logic of the assembly system.

	Field	Example Values	Type
Header	Serial_No	2563	int
	OrderID	24535	int
Process 1	Process_Name	"Drill"	String
	Parameters	*"Drill holes XY: (35,6;23,5)"*	String
	Feedback	*"Motor temp:47C"*	String (xml)
	Excecuted	yes/no	bool
	Succesful:	yes/no	bool
	Skills_needed	"Drilling"	String
		"Dispenser:Part ID"	String
		"Fixture: insert"	String
		"Mover"	String
		"AssemblyParameter"	String
	ResultingPartID:	1253	int
	Next process_Name:	"Vision1"	
	Self starting:	yes/no	bool
	Start_timestamp:	"2019.10.91 15:56:53"	DATETIME
	End_timestamp:		DATETIME
	PartID1_consumed	253	int
	Parts no. Consumed	2	int
Process 2	Process_Name	"Vision_orient"	String
	Parameters	*"check orientation"*	String
	Feedback	*"Orientation:"*	String (xml)
		...	
		...	

Fig. 4 Excerpt from a product recipe.

Apart from the process parameters, the process entity contains other fields, such as a placeholder for feedback from the agent (eg. a dump of sensor values for future data analysis), as shown in figure 4. As an alternative to the placeholder, timestamps could be created traceability of the order throughout the production. Together the

fields contain all the information needed to control and monitor any type of manufacturing process that can be added to the Smart Lab.

5. Software Architecture

The ontology described in section 4 is in practice realized by using the infrastructure already built into the Smart Lab. The recipes are managed by a Manufacturing Execution System (MES) running on a personal computer, which via the common connection bus is connected to the PLC of each basic module(workstation). This in turn connects to the process modules, as shown in figure 5. All the information such as the process parameters and the process feedback is passed between the parties via. a TCP-IP connection.

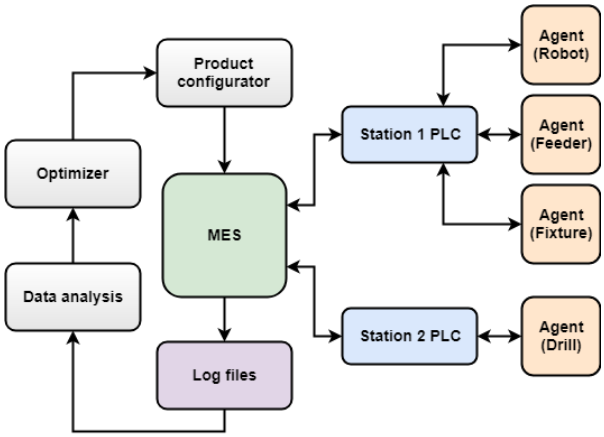

Fig. 5 Overall software architecture.

Product configurator

The recipe is envisioned to be generated by the *product configurator*, which based on the customers wishes, assembles together a recipe for the product. As the process parameters for each skill are vendor-specific, the product configurator receives the process parameter string for a given machine setting (as a plugin), accompanying the given equipment module, as illustrated in figure 6. This approach implies that each configuration of the product can be created by assembling together a list of all the necessary processes, in contrast to the current setup, which stores all the possible product combinations in a database.

Introducing a new process or an update would then only require changing a single plugin, instead of finding and updating all the affected combinations in the database. As the processes have an optional field for the parts consumed, the finished recipe can then act as a Manufacturing Bill of Materials (mBOM), in connection with an ERP system, the configurator can therefore be used for checking the availability of all parts needed in the recipe.

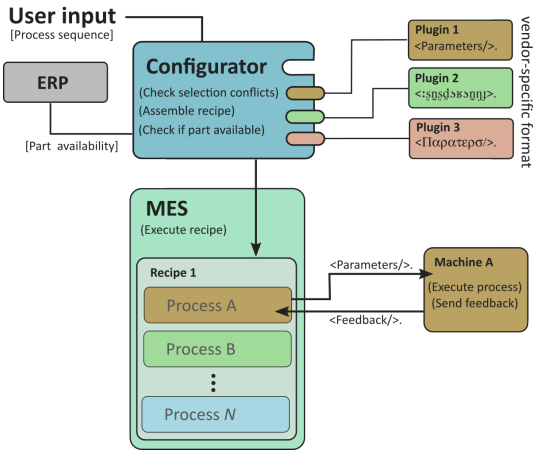

Fig. 6 Interaction between the product configurator and the software plugins.

MES system

The MES-system is responsible for executing and managing the order recipes, acting as a server for the process agents, it handles data requests to the internal database of pending and executed recipes.

Additionally the MES system is responsible for routing the carriers on the conveyor system. At system start-up the MES learns the physical layout of the plant (currently via. manual input). When a carrier arrives at a conveyor branch, the basic module PLC responsible for operating the conveyors reads the "next_skill" field on the carrier RFID and requests a direction. The MES system, then guides the branch module, knowing the direction which will lead to the desired process.

Post-processing and optimization

After the recipe for a given order has been executed the "Feedback" fields and time stamps from each process can be collated in a data log of all relevant information, such as results from the automatic quality control, sensor values from the machine during process execution etc., creating a data trace for each phone produced.

A data analysis tool can then be used to analyze the log files and improve future product recipes, for example by fine tuning the length of the release timer, described in figure 11 of section 5.4, in order to match the merging of the two process streams.

5.1 System operation

In order to operate the manufacturing system, all process modules are first physically mounted/rearranged on the Smart Lab and connected to the network sockets. A zero-configuration network routine is then performed to establish a connection between the parties and uncovering the network structure shown on figure 5. Next, the MES system requests each agent to identify their skill that they offer, and compares the list of available skills with the skills required in the pending recipes. As the MES system also knows physical layout of the plant, using the network topology, it is then able to create a map of were the modules are located, which is later used for directing the phone carriers.
Once the availability of all skills has been confirmed, the recipe is put into production by the MES directly ordering the PLC with of the first process module to start execution.

5.2 Executing a simple skill

After the first process has been triggered directly from the MES, the execution of each subsequent process is de-centrally initiated by the part carrier.

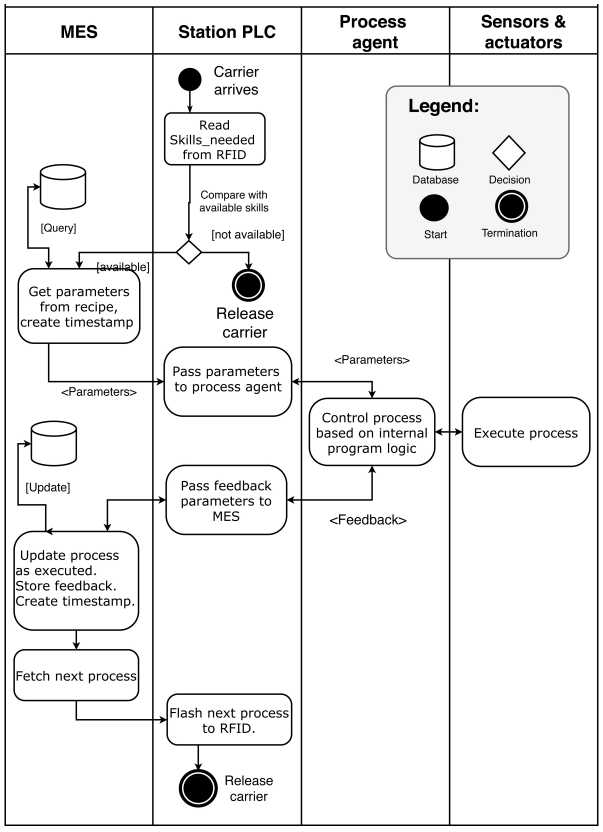

Fig. 7 Execution of a single-skill process

The diagram on figure 7 shows the execution of a simple process involving only one skill agent:

1) The part carrier arrives at the conveyor stopper of the basic linear module. The PLC from the basic module reads the order number and "next process_ skills" field on the RFID tag of the carrier, and compares it with its internal list of skill agents connected to it. If the skill needed is present, the carrier is stopped.
2) Using the serial number, the PLC requests the process parameters for the operation from the MES system, and passes them on to the relevant process module. The MES system creates a timestamp for the process start in the product recipe.
3) Based on the passed process parameters, the process module micro-controller executes the operation. It passes back a confirmation, together with the process feedback. The PLC passes those back to the MES system, which updates the process in the recipe as executed, and returns the skills needed for the next operation in the recipe.
4) The PLC flashes the next operation onto the RFID chip, and, if the next operation is not on the same station, releases the carrier.

This arrangement is very similar to what is currently implemented at the Smart Lab, with the exception of the vendor-independent process control the feedback information being collected and stored.

5.3 Executing a multi-agent assembly

Building on the same principles the process can be extended to involve multiple skill agents, with the most relevant multi-agent process being the assembly, as both the fuses and the PCB are inserted using a robot, as described in section 1. In a generalized approach, an assembly involves three parties shown on figure 8: A *feeder* being the module dispensing the needed part, a *fixture* being the intended target the part is to be inserted to and a *mover* being an agent (whether human or robotic) capable of performing the assembly. Each of them is represented as a skill, possessing information about their own position and state needed to complete the assembly. As the assembly requires coordination between the parties, an *orchestrator* entity is introduced, being a software agent running on the Linear module PLC, responsible for gathering the information and generating a motion path for the robot.

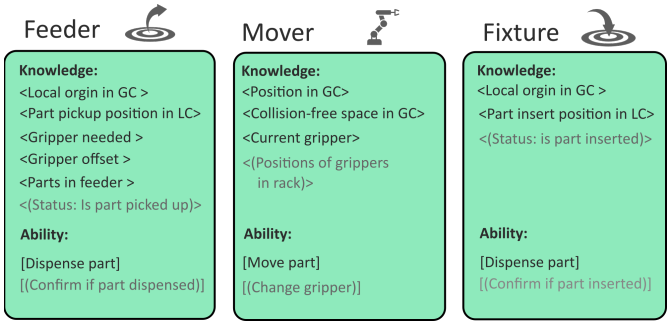

Fig. 8 Semantic model of the 3 assembly agents. Fields embraced by parenthesis "(..)" are optional.

The UML diagram on figure, 11, shows the execution logic for an assembly:

1) The first steps in the assembly process are identical to steps 1 and 2 for a simple process, described in figure 7 and section 5.2. But additionally, apart from the skills needed to execute the assembly, the process recipe can optionally specify additional information needed for the assembly that has to be requested from the MES system. (Eg. in the case of the Smart Lab, the orientation of the lower housing on the carrier, which is inspected by a vision system prior to carrier entering the robot cell.) The process parameters sent back from the MES system specify the part ID of the part to be picked from the feeder, and the *fixture slot* ie. the position in the fixture the part should be inserted into. (for example, the lower fuse holder in the PCB is a fixture slot)
2) The process parameters and the additional information are further passed to the assembly orchestrator within the PLC. The orchestrator connects to the agents and awaits the feeder validation of the Part ID (is the part available in the feeder). The feeder specifies the name and tool offset for the gripper in respect to the robot flange. The availability of the fixture slot is also validated as parallel routine (is the requested slot an existing, free slot in the fixture).
3) If both replies are positive, the orchestrator requests the process module's anchoring coordinates and it's pick-up point in respect to a global coordinate system of the assembly station (i.e. a stopper or a corner of a table). This establishes the points for the trajectories used the pick-and-place routine as illustrated in figure 9.

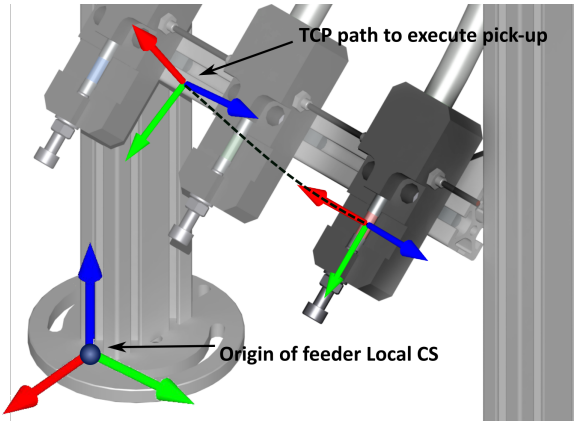

Fig. 9 Illustration of a pick-up path being specified by the fuse dispenser, guiding the robot gripper

4) The robot has a known collision-free space, as shown on figure 10 in which it can move between the end-paths specified by the other modules. Based on the information gathered, the orchestrator generates pick-up point coordinates for the robot to pick up the part from the feeder and place it in the assembly, and asks the robot to execute it, then confirms the part has been dispensed and inserted successfully.

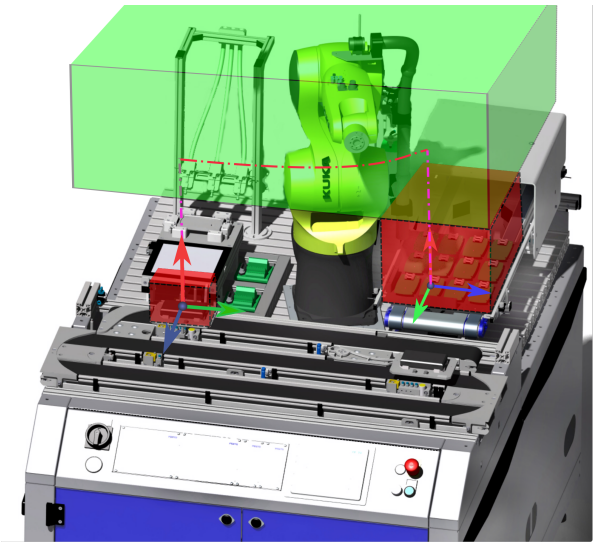

Fig. 10 The assembly station, as seen by the assembly orchestrator. The pick-up and place locations are specified by the feeder and fixture respectively, while the orchestrator connects the two endpoints in the known collision-free space marked in green.

5) Before the carrier leaves, similar to the step 4 in section 5.2, the fixture receives the part ID of the new assembly. This enables the sub-assembly to be used as a part in another assembly process, where the fixture becomes a feeder for the next process. (For example: The fuses are

inserted into the PCB's waiting on the PCB tray, where the PCB tray becomes a feeder when the new PCB/fuse sub-assembly is inserted into the housing.)

By using this set-up, the robotic assembly becomes an instance of a generic pick-and place process, independent of the specifics of the robot cell application. As all the information about the fixtures and feeder is carried within the relevant modules, modifying the robot cell would only require changing the modules and updating their local origin positions in the global coordinate system. This could be achieved by using a set of defined mounting points on the slot table (i.e. a stopper) and by using a hooking mechanism for accurately positioning the additional robots. The sophistication of the fixture and feeder can also vary, ranging from the simple fuse feeder, shown on figure 9, to more advanced, vision-based solutions, depending on the application.

5.4 Multiple production lines

The processes in the existing assembly line configuration, can only be executed sequentially. To increase the flexibility and speed of the production system, the project recipe can be configured to include a parallel stream of processes, as shown on figure 12. The execution of the processes in the parallel branch follows the same methodology as described in section 5.2. The first process is invoked directly by the MES system, and is triggered when a another process in the main branch starts. At the end of the parallel production stream the sub-assembly is put in a waiting pool, until the correct carrier arrives at the main assembly line. It is assumed that two production lines are always merged by an assembly operation.

A waiting timer can be inserted between the process trigger and execution start, to better time the rendezvous of the two parts. An example of this being applied in the Smart Lab would be the 3rd configuration described in section 1, where the fuses are inserted into the PCB's on a side branch of the production line and assembled into the main carrier.

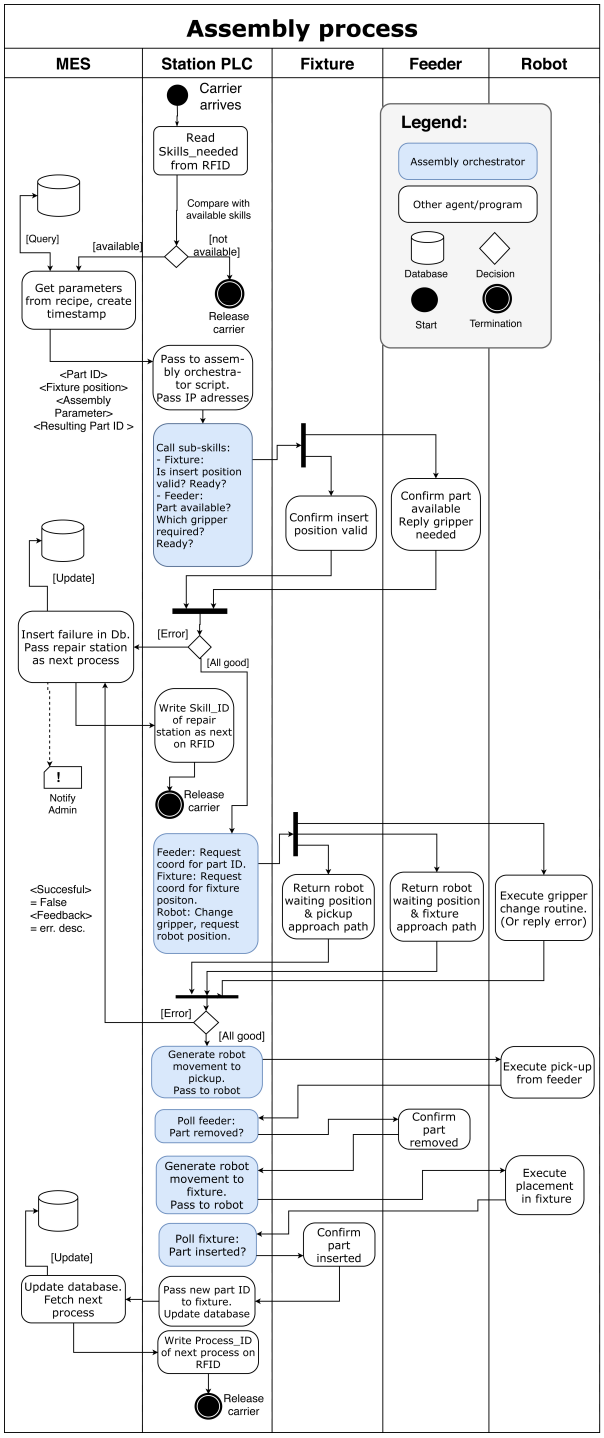

Fig. 11 UML activity diagram for the assembly sequence.

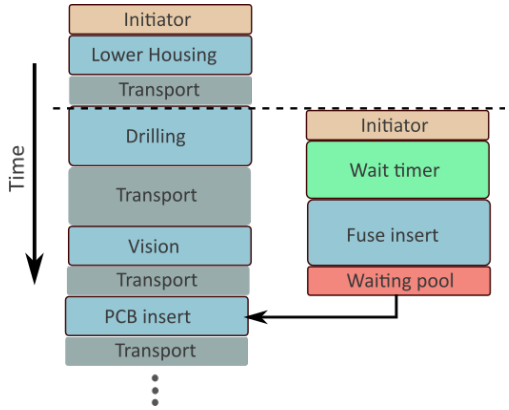

Fig. 12 The two parallel processes - Fuse insertion into the PCB

6. Conclusive results and future work

The framework described in this paper has been implemented in a simple technology-demonstrator setup, utilizing 3 Raspberry Pi single-board computers, connected in a cluster. The Raspberries are communicating via a direct TCP-IP connection, as shown in figure 13. One Raspberry Pi is acting as the MES system and it hosts a server for the client Raspberry Pi which is emulating the station PLC. Order recipes, in the form of an SQL database, are also stored on the Raspberry Pi which is acting as the MES system, where the database is accessed utilizing a Python client to handle the SQL queries. The remaining Pi is emulating the agent, which acts as a client to the second server hosted on PLC emulator.

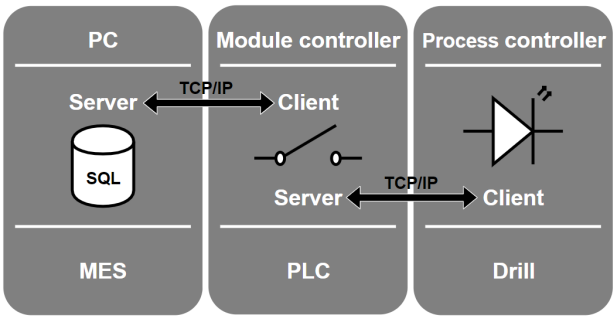

Fig. 13 Prototype setup scheme

A future expansion of the current project could be to implement it using the OPC UA communication protocol, which is intended for IoT devices and provides communication between field devices and upper control levels. The protocol offers platform-independent web-services-based data model, allowing to create OPC UA "Programs" (semantic models) that expose the data from a device to the system [12] [13]. Furthermore, the protocol offers a number of other functionalities, such as a *Discovery Server* feature allowing newly connected clients to automatically connect themselves to the network, which would simplify addition of new process modules in the Smart Lab, or a *Historian Server* type allowing to collect and store data from multiple OPC UA servers into one central location [13].

Acknowledgement

The authors of this work gratefully acknowledge Grundfos for sponsoring the 7th MechMan symposium.

References

[1] H. A. ElMaraghy, Changeable and Reconfigurable Manufacturing Systems. Springer, 2009.

[2] M. Onori, N. Lohse, J. Barata, and C. Hanisch, "The ideas project: Plug & produce at shop-floor level," Assembly Automation, vol. 32, pp. 124–134, 2012.

[3] P. Ferreira and N. Lohse, "Configuration model for evolvable assembly systems," 4th CIRP Conference on Assembly Technology and Systems, 05 2012.

[4] R. F. Babiceanu and F. F. Chen, "Development and applications of holonic manufacturing systems: A survey," Journal of Intelligent Manufacturing, vol. 17, no. 1, pp. 111–131, 2006.

[5] Z. M.Bi, S. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: the state of the art," International Journal of Production Research, vol. 46, no. 4, pp. 967–992, 2008.

[6] Z. M.Bi, S. Lang, W. Shen, and L. Wang, "Current status of reconfigurable assembly systems," International Journal of Manufacturing Research, vol. 2, no. 3, pp. 303–328, 2007.

[7] OpenMOS, "Aim & objectives."

[8] N. Lohse and P. Ferreira, "Deliverable d3.7: Common semantic model," tech. rep., OpenMOS project, 2018.

[9] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," tech. rep., Knowledge Systems Laboratory, Stanford University., 1993.

[10] A.-B. M.Salem and M. Alfonse, "Ontology versus semantic networks for medical knowledge representation," 2th WSEAS International Conference on Computers, 2008.

[11] G. Pritschow and K-H.Wurst, "Control of reconfigurable machine tools," in Changeable and Reconfigurable Manufacturing Systems (H. E.

Maraghy, ed.), ch. 4, pp. 266–290, Springer, 2009.

[12] K. Dorofeev and A. Zoitl, "Skill-based engineering approach using opc ua programs," 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), pp. 1098–1103, 2018.

[13] O. U. Foundation, "Opc unified architecture specification," tech. rep.