Deployment of a Distributed Multi-Agent Architecture for Transformable Assembly

Jack C. Chaplin(✉) and Svetan Ratchev

Institute for Advanced Manufacturing, Advanced Manufacturing Building, Jubilee Campus, University of Nottingham, Nottingham NG7 2GX, UK
jack.chaplin@nottingham.ac.uk

Abstract. Industry 4.0 represents a new philosophy in manufacturing systems, based on networked, intelligent, and cooperative resources. This revolution is necessary to make the cost-effective production of batch-size-of-one customised items in high-value manufacturing domains such as aerospace a reality. However, there exist large numbers of legacy production cells which generate value for enterprises which would ideally become part of a future manufacturing system, but which lack the necessary computational or networking capabilities. This is especially important in the case of small to medium enterprises, where Industry 4.0 is perceived as an expensive endeavour out of reach due to cost. There is a requirement for Industry 4.0 to be brought to existing legacy production cells in a cost effective and standards-compliant manner. This paper describes the technical implementation of an Evolvable Assembly Systems deployment onto an existing legacy manufacturing cell, describing the concepts and technical specifics of how to interface a software-based multi-agent system with real manufacturing hardware, and demonstrates how it is possible to make a transformable manufacturing cell which is compliant to the Industry 4.0 ideals in a cost-effective and expedient manner.

Keywords: Industry 4.0 · Evolvable Assembly Systems · Smart manufacturing

1 Introduction

Higher product complexity in terms of physically complex designs and assemblies with stringent tolerance requirements, demand for customer-unique customisation, and increasingly smart products with bespoke software installed at assembly-time is a significant challenge for modern and future manufacturing. Exacerbating this situation, products are changing to meet customer demands increasingly quickly, reducing the time available to develop an assembly process for a product [1].

There are numerous national and international initiatives to address the challenges of managing the data generated by such complex manufacturing processes, so more effective decisions can be made quicker. These include Industry 4.0 [2] and the Industry 4.0 reference architecture RAMI4.0 [3], and the Industrial Internet Consortium and their Industrial Internet Reference Architecture (IIRA) [4].

There is a need for distributed intelligent control that can mitigate the exponential rise in the number of communication interconnections between increasing numbers of networked manufacturing resources and sensors, which would enable production lines to be agile in response to new product variants and robust to system failure.

In addition, with individually customised products becoming increasingly important in the manufacturing economies of high-technology, high-wage cost regions, there is a requirement to track and log data associated with every single product created, to ensure full traceability when a product has been assembled with a unique assembly strategy. This is particularly important in highly regulated domains such as aerospace, pharmaceuticals, and construction.

Many modern production systems utilise Programmable Logic Controllers (PLCs) to control manufacturing resources, and which possess the ability to exchange and process information. However, these can often only communicate via vendor-specific protocols, and program execution may be constrained to IEC 61131-3 programming languages [5], limiting the opportunity for more complex algorithms. In addition, many legacy production resources still provide value to their operators, but these may lack any computational capacity or utilise obsolete communication methods if they possess any communication methods at all. It is important that the fourth industrial revolution does not pass over small to medium enterprises (SMEs), who may have higher proportions of legacy equipment because they cannot afford to purchase expensive new resources or controllers, or lack the time and expertise to invest in manufacturing digitisation [6]. Hence, a solution must be found to include legacy resources in any smart manufacturing system in a homogeneous, simple, and low-cost way.

This requirement for an open, accessible, and low-cost solution to the digitalisation of manufacturing processes would enable the implementation of system-wide intelligence, and a strategy for sharing system-generated data to allow companies to participate in the fourth industrial revolution. The system must facilitate the dynamic checking of the manufacturability of new products to ensure the system can keep up with the increasing pace of product change, and facilitate transformable manufacturing cells able to evolve to match changing product requirements. Information on the manufacturing process must be logged to ensure traceability of product data.

There are existing architectures for distributed system-wide intelligence that tackle the above problems based on multi-agent systems [7–9]. These include the Product-Resource-Order-Staff Architecture (PROSA) [10], the Adaptive Holonic Control Architecture for Distributed Manufacturing Systems (ADACOR) [11], and Evolvable Assembly Systems (EAS) [12]. However, how these software architectures are actually deployed on to existing legacy manufacturing lines remains an open challenge.

This paper details the implementation of a technical solution for deploying a distributed multi-agent manufacturing intelligence architecture (specifically Evolvable Assembly Systems) on to an existing legacy robotic manufacturing cell which lacks inherent Industry 4.0 features. It does this in a standards-compliant, low-cost way which would be applicable to SMEs which want to take advantage of Industry 4.0 principles, but cannot risk the high expenditure required to replace existing production cells.

2 Scenario and Current Limitations

To demonstrate the approach described in this paper, we present a scenario using aerospace components on a robotic assembly cell. The cell forms part of a larger assembly line – the Future Automated Aerospace Assembly Demonstrator (FA3D) – and prepares rib components for inclusion in a wing or fuselage assembly by applying sealant and/or scanning the part with a line scanner to check part accuracy and the quality of the sealant application.

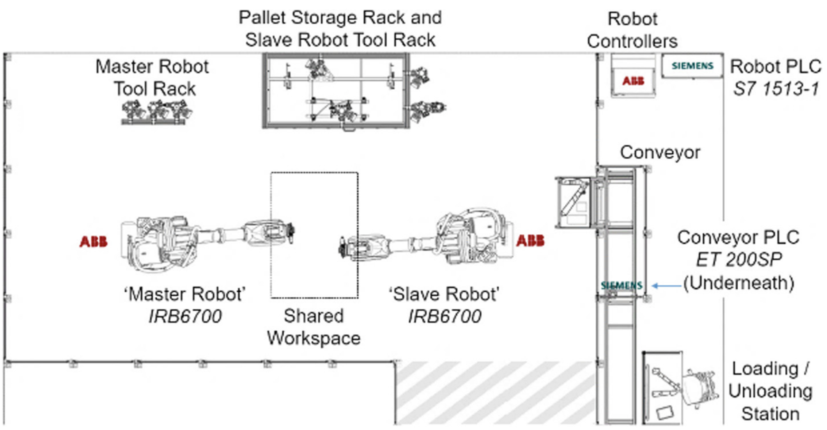
This cell has two primary challenges that must be met for which the current control method is inadequate. The first challenge is that the FA3D assembles batch-size-of-one products, where every item being assembled is unique and must be treated differently. The larger assembly line is required to produce aircraft components for high-complexity, highly-customisable, and low-volume airframes while keeping costs low, and as such each assembled airframe can be considered to be unique. Additionally, the high precisions required of the final assembly require every part to be analysed for deviations from specification, and the assembly process altered to compensate. As a result, seemingly identical components being handled in the cell must be identifiable and able to be treated differently as the quality of supply varies.

The second primary challenge for the cell is the logging of data. In highly regulated industries such as aerospace, pharmaceuticals, or construction, logging data is essential for tracing the root causes of problems, analysing data to identify problems before they occur, and providing accountability for failures. This is especially important for batch-size-of-one manufacturing, where every product is assembled in a unique manner which may be determined automatically by intelligent manufacturing systems. The data must be interrogable to understand why decisions were made in the event of failure, and to understand the behaviour of the system over time.

The demonstration cell operates on rib components for aerospace assemblies. The ribs are placed in pallets, which can then be moved around the cell. The cell utilises two ABB IRB6700-150-320 robotic arms to perform operations on the rib components. The robots are set-up in a master-slave configuration, with a single Siemens S7 1513-1 PN PLC controlling both. Each robot has a separate tool changing rack. The master robot has access to a proxy sealant applicator (to facilitate testing), and a Micro Epsilon scanCONTROL compact 2900-50/BL laser line scanner. The slave robot has access to a bespoke pallet gripper and a bespoke rib gripper, allowing it to pick up pallets or pick ribs out of pallets. To one side is a storage rack, where pallets with or without ribs can be stored and retrieved.

The cell has a loading/unloading area for loading pallets and ribs onto a conveyor system. Loaded pallets move into the slave robot's working area, allowing the pallet or rib to be picked up. The conveyor system is to be expanded to allow complete ribs in pallets to be dispatched to the second cell to be assembled in a larger aerospace assembly. The conveyor system is controlled by a Siemens SIMATIC ET 200SP Open Controller with software-based PLC functionality (Fig. 1).

The cell in its current state has several technical limitations. With the expansion to the conveyor system planned to link the cell to the wider FA3D production line, these



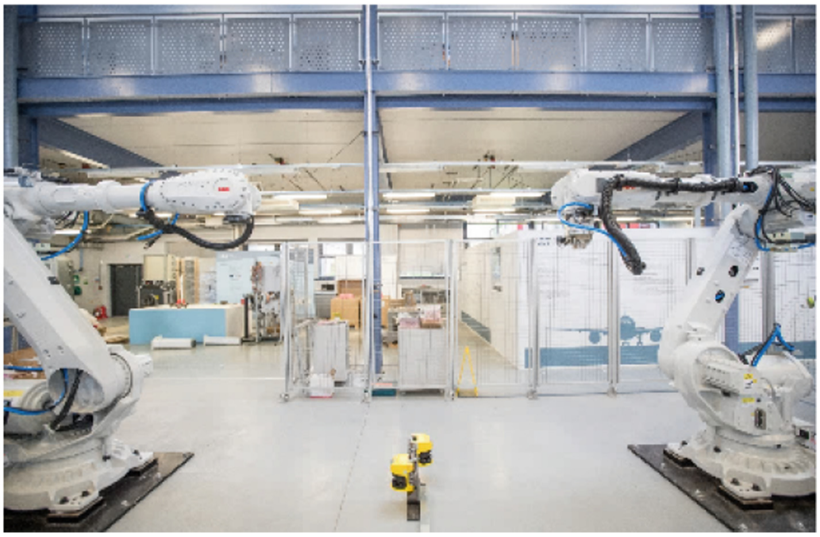


Fig. 1. (Top) Schematic layout of the demonstrator cell. (Bottom) View of the cell from in front of the storage rack, showing the two robots.

limitations must be overcome if the cell is able to contribute to high-precision, batch-size-of-one products.

The first limitation is that the demonstration cell has no networking between its components; the PLCs for the robots and for the conveyor system are not connected, and instead rely on user interfaces with which a worker can execute several pre-specified actions manually.

The second limitation is that it does not have any system-wide control that allows for automated production of parts. For batch-size-of-one products, this approach is insufficient as the worker must understand the production plan for a potentially complex part sufficiently to correctly pick the order of operations, as well as to know when to perform any manual operations such as loading/unloading parts. Additionally, with no networking, the system cannot communicate with the wider production line, which is a significant problem if this cell is to be integrated with the FA3D.

The third limitation is that data collected by the system is transitory and not stored, which does not allow for retrospective analysis of cell performance or identification of errors. Additionally, data generated by the laser line scanner is saved as a file, which must then be manually transferred to the FA3D via USB drive. To remove this manual step, it would be preferable for data to be transferred automatically to the appropriate resources.

The fourth limitation is that the actions which can be performed by the robots and the conveyor system are fixed. The commissioning of the cell included programming the PLCs and specifying the actions performable by the robots, and changing these risks violating the safety certification of the cell necessary to have the cell open for manual worker access.

3 Enabling Technologies for Low-Cost Industry 4.0 Implementation

To address these limitations and upgrade the cell, we propose the use of technologies which directly tackle the issues of networking, system-wide intelligence, and data logging, and also mitigate the issue of fixed actions which are inherent to including legacy production cells into Industry 4.0-compliant manufacturing lines. These technologies and approaches are discussed here.

3.1 Data Distribution Services

Increasing the number of devices within manufacturing production lines, combined with a requirement for devices to talk to other devices in the line requires an exponential number of interconnects. Though service-orientated architectures are an improvement on existing point-to-point solutions, this still requires a client/server relationship. Instead, the approach taken by the Industrial Internet Consortium [4] to decouple industrial manufacturing as much as possible is the use of Data Distribution Services (DDS) [13], which is a standard administered by the Object Management Group (OMG).

DDS is a networking middleware designed to decouple the origins and consumers of data within a specified domain. Nodes in the DDS that produce data are called publishers, and publish pieces of information (called samples) to topics. A publisher can publish data without needing to know what (if any) nodes will consume this data. Nodes which consume data are called subscribers, and subscribe to topics to receive samples published there. Similarly, a subscriber can subscribe to topics without needing to know the origin node of the data.

The result is a data-driven communication system, where the most important element – the data – drives the operation of the manufacturing system. Nodes in the system do not communicate directly, but instead form a shared system context where a single canonical view of the state of the system is distributed between the nodes, and from which nodes can selectively subscribe to information relevant to them.

The lack of a communication broker, or direct interconnects between resources in the system serves two important roles in the context of a manufacturing system. Firstly, there is no single point of failure and data is replicated by the middleware between nodes (in most DDS implementations), leaving a robust system that can continue execution even if nodes fail. Secondly, with no explicit network structure, the system can be reconfigured and nodes added or removed without having to alter the networking configuration. Additionally, if the networking uses wireless methods (as with this demonstration), there is no requirement to alter the physical networking infrastructure either.

3.2 Embedded Computers

With a data-driven approach, there is no direct communication between resources in the manufacturing system. As a result, there is no single orchestrator that instructs each resource as to what to do and when to do it, in contrast to traditional monolithic architecture. Instead, each resource must be smart enough to analyse the state of the system in the shared system context and make decisions as to what to do. For this, every resource in the system requires a degree of computing power, which is not something that can be assumed for legacy systems. Neither can it be assumed that resources with computing abilities (such as those with Programmable Logic Controllers) are provided as open by their vendors and hence capable of running the necessary decision making algorithms or DDS implementation.

To solve this problem, the approach given here uses embedded computers to handle decision making and communication, as this allows for the execution of general-purpose code and use of communication standards which might not otherwise by available. This approach removes vendor-specific implementation considerations at the earliest possible stage, and transforms all control and information into homogeneous and open methods. This does require an interface between the embedded computer and the resource controller, which may be a modern PLC with interface APIs, a legacy controller with no consideration for external interfacing, or a human worker which will require a smart or wearable device to communicate information.

Embedded computers also provide local data storage capacity, and the ability to connect to larger capacity database, enabling the storage of large quantities of operational data that would otherwise be lost, which could be pre-processed and analysed at the edges of the system to extract events and filter out unneeded data.

Combined with the data distribution services, the use of embedded computer solves the challenges of networking and lost data. However, the challenges of system-level intelligence and fixed actions remain. To solve this limitation, we utilise intelligent agents.

3.3 Intelligent Agents

Each module in an Industry 4.0 enabled cell requires localised intelligence to make decisions, be context aware, and share information with the rest of the production line. These intelligences represent the production resources which carry out operations on parts in line with their capabilities.

To implement localised intelligence, we utilise Belief-Desire-Intention (BDI) [14, 15] intelligent agents (or simply ‘agents’) [7–9] which are independent, self-contained software entities which use the BDI model of planning, allowing a decoupling of the agent’s perceptions of the world, the selection of plans, and the execution of plans. These agents execute on the embedded computers, and are the interface with the wider production line, publishing and subscribing to the shared system context, and making decisions based on the state of the cell as to when to execute capabilities. Upon these agents, we implement the Evolvable Assembly Systems architecture.

4 Technical Implementation of Evolvable Assembly Systems

The three key technologies of DDS, embedded computers, and intelligent agents can be used together to create an implementation of a smart Industry 4.0-compatible manufacturing system. Combined with the data distribution services, the use of embedded computers solves the challenges of networking and lost data. However, the challenge of system-wide intelligence remains despite the use of intelligent agents – agents are a general purpose solution and not tailored to manufacturing cells. To solve this challenge, we employ Evolvable Assembly Systems.

Evolvable Assembly Systems [7, 8] is a manufacturing paradigm defining transformable, responsive production lines for effectively producing low-volume products. It focuses on the use of multiple independent and modular components to allow for rapidly reconfigurable and transformable production cells. The use of intelligent agents to execute the Evolvable Assembly System paradigm to make decisions as to what actions to perform based on the state of the system as per the shared system context satisfies the requirement for system-wide intelligence, as this enables batch-size-of-one production where each step in manufacturing is defined in a recipe. Our software implementation of EAS is detailed in [12, 16], and utilises agents to implement distributed intelligence in manufacturing systems, the automated checking of the manufacturability of submitted batch-size-of-one product recipes against the capabilities of a production cell or line, and allocation of requirements to manufacturing resources. However, the technical challenges remain of how the agents are deployed, and how the determined requirements are automatically executed on manufacturing resources. This is solved with the approach detailed in this section, resulting in a functional demonstrator of an Evolvable Assembly systems deployment.

4.1 Embedded Computers and Smart Devices

To implement our approach, we used embedded computers. A wide variety of embedded computers are available, ranging from the basic Raspberry Pi Zero W [17], to more fully featured products such as the Siemens SIMATIC IOT2000 Intelligent Gateway series [18]. The exact model used will depend on the interface requirements with the resource controller. As our implementation is hardware agnostic, a mixture of embedded computer models could be used.

In our case, the Siemens SIMATIC S7 1513-1 PN PLC and SIMATIC ET 200SP Open Controller offer interfacing via Ethernet connections, so any embedded computer would require an Ethernet port. Our code, including BDI agents, networking middleware, and manufacturability analysis implementation described in [16], is implemented in Java so any model chosen must support this. Communication via the data-centric shared system context between nodes is also required. As the demonstrator used in this example is not physically reconfigurable, wireless networking has less merit than in other more modular reconfigurable demonstrators such as those presented in [12, 16]. However, wireless communication is important when a human worker is involved in the process as with this demonstrator, as the human will be mobile and utilising a smart device to interface with the system. Additionally, there are few models of embedded computer with two Ethernet ports (as one is required to connect to the PLC), and so wireless networking was determined as preferential over wired technologies.

To satisfy these requirements, we selected one of the most widely available single-board computers to use as our standard model of embedded computer – the Raspberry Pi 3 Model B [19]. The Raspberry Pi is globally available, low cost, open, and flexible, and features all the connectivity we require, including in-built wireless connectivity. It is powered via a micro-USB 2.5A power source.

Though more computationally powerful or more fully featured single board computers exist, the Raspberry Pi is useful as a baseline for demonstrating this approach, as a successful implementation on a Raspberry Pi would also function on other more powerful and more expensive models. These embedded computers enable the execution of intelligent agents which communicate and cooperate, giving the demonstration cell system-wide intelligence and context awareness as required.

Human workers in the production line also require interfacing to the wider system intelligence. This is achieved through the use of mobile smart devices. For now, we utilise Lenovo X260 Ultrabooks, though future work will include the deployment of the interfaces to low-cost tablet computers (Fig. 2).

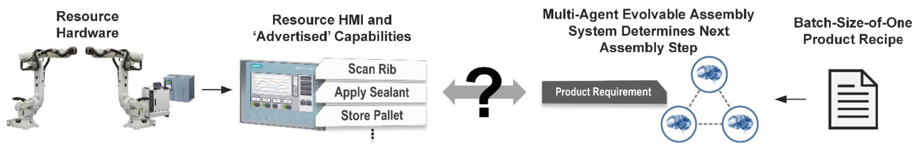

Resource Hardware
Resource HMI and 'Advertised' Capabilities
Scan Rib
Apply Sealant
Store Pallet
?
Multi-Agent Evolvable Assembly System Determines Next Assembly Step
Product Requirement
Batch-Size-of-One Product Recipe

Fig. 2. A legacy production resource can be considered to be advertising its capabilities via a user interface. A smart manufacturing system such as an Evolvable Assembly System wishes to instantiate these capabilities. But how can the intelligent agents actually interface with the resource?

4.2 Interfaces with Resources

The intelligent agents executing on the embedded computers and smart devices require interfaces with the resources (or human workers), enabling the requesting of capability instantiation and collection of data.

This approach suggests use of service-orientated architecture such as OPC Unified Architecture [20] where the resource advertises services (i.e. the instantiation of capabilities) and the embedded computer acts as the client to select these. However, retrofitting these capabilities into existing production lines (such as this demonstration cell) does not necessarily allow for such idealised solutions. PLC code may have been tested and proven correct for years prior to the decision being made to include Industry 4.0 technologies, and stakeholders may not want to risk sweeping changes to PLC execution. Additionally, some legacy resources may not be able to execute OPC UA (or similar) at all. An alternative solution must be found.

As the demonstration cell was designed to be used manually, the PLCs to control the robots and to control the conveyor system both have user interfaces by which a worker can instantiate capabilities by pressing virtual buttons on a touch screen interface. Though not designed with service-orientation in mind, a user interface essentially allows the resource to advertise capabilities in the form of buttons or other UI elements, and to receive requests from a client in the form of button presses.

To leverage this indirect service orientation, the embedded computers must be able to simulate a button press on the user interface, which enables automated execution of PLC code without modifying the PLC code itself. The precise manner in which this is achieved will necessarily vary depending on the user interface and vendor in question. Presented here is the method used for both the Siemens SIMATIC S7 1513-1 PN PLC and SIMATIC ET 200SP Open Controller used for the robots and conveyor system respectively.

User interfaces for Siemens PLCs are created using SIMATIC WinCC [21]. This executes on the PLC, but in a separate memory space to the main PLC code. Buttons are created and have actions associated with them when the button is pressed; the general pattern is for a bit in memory to be set when the button is depressed and reset when the button is released. Code in the PLC checks the state of the bit associated with the button and performs actions if the bit is set. Despite both the S7 and Open Controller PLCs having different internal architectures, they both follow this pattern (with minor deviations as both were commissioned by different organisations).

S7Connector is an open source Java library for interfacing with S7 PLCs via Ethernet [22]. As the ET 200SP is based on an S7-1500 CPU, both PLCs are compatible with this approach. It allows for reading and writing values to the memory of the PLC via a Java program executing on a connected device. This permits the connected device to alter the bit that is set by pressing a button on the user interface, simulating a button press.

One limitation of this approach is that it requires knowledge of the memory location of the bit connected to the button on the user interface, which in turn means access to the original project files for the PLC. Interfacing with a user interface via an embedded computer without access to the source for the PLC program would be challenging, and other solutions may be required, such as having the resource embedded computer using its own interface on a smart device to instruct a worker to press the required button.

With the same approach, any data made available to the user interface (resource status or data generated by the line scanner for example) can be retrieved in the same manner by reading the variable which controls the UI element display. This allows the embedded computer to know what the attached resource is doing, and read (and optionally publish) data it generates.

Human workers are also perceived in Evolvable Assembly Systems as manufacturing resources, and hence require an interface between the smart device on which code executes and the worker themselves. This takes the form of a user interface, instructing the operator what tasks to perform at a given time, and offering the user the ability to specify the completion of work (Figs. 3 and 4).

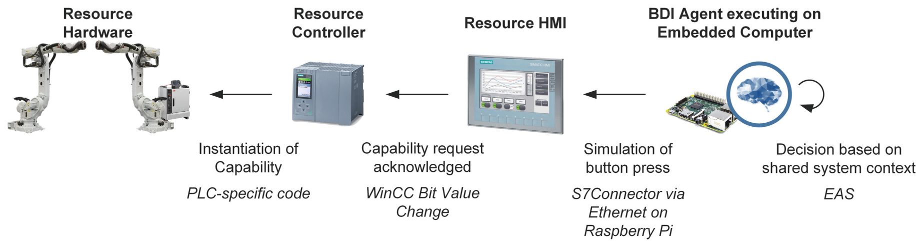
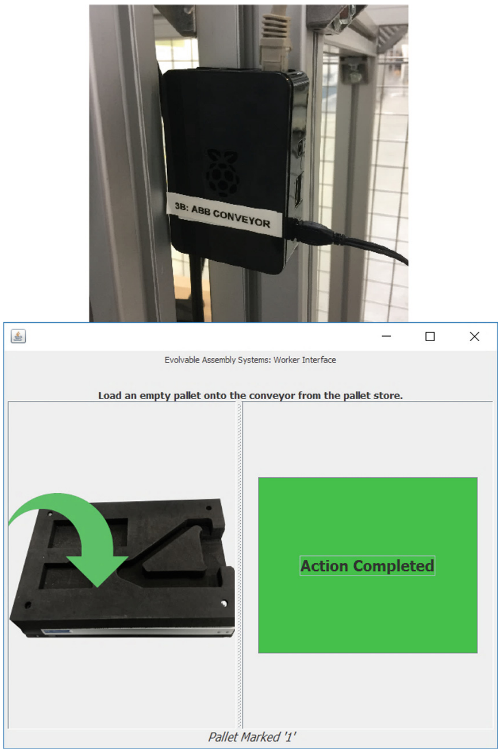

Resource Hardware
Resource Controller
Resource HMI
BDI Agent executing on Embedded Computer
Instantiation of Capability
PLC-specific code
Capability request acknowledged
WinCC Bit Value Change
Simulation of button press
S7Connector via Ethernet on Raspberry Pi
Decision based on shared system context
EAS

Fig. 3. The communication stack for an agent to execute a capability on a resource, using the example of the master/slave robots.



Load an empty pallet onto the conveyor from the pallet store.

Fig. 4. (Left) A Raspberry Pi 3 Model B is networked to the conveyor PLC. (Right) A worker is instructed to load a pallet onto the conveyor system by the worker resource interface.

4.3 Networking and Data Collection

Networking resources in the production cell allows for data to be published and subscribed to via DDS and the shared system context. As the Raspberry Pi 3 Model Bs and Lenovo X260s feature built-in Wi-Fi, and as the Raspberry Pi Ethernet ports are used to interface with the PLCs, a wireless network was created using a TP-LINK TD-W8970 router. The wireless networking allows for the mobility of the human workers with laptops (and later, tablets or other smart devices), and simplifies system reconfiguration. To operate using a data-centric model, we use DDS. For this implementation, we use RTI's Connext DDS Professional [23], which is a fully featured and compliant implementation of the OMG DDS standard [13]. It has Java libraries and native support for Raspberry Pi single board computers.

With advances in data storage and data-centric manufacturing systems, it's now possible to store all data on a product being assembled, including a digital 'black box' of why decisions were made, to facilitate the tracing of errors and verification of system behaviour. We call this ProductDNA, and it includes all data generated by the involved resources, any metrology data produced including environmental data, tolerances and deviations, and system decisions made and the reasons why.

To achieve this, an additional ProductDNA agent is added to the manufacturing cell, executing on a Dell Precision 7510 laptop. This agent, like other agents in an Evolvable Assembly System, can subscribe to topics to receive data. To gather all possible data, it subscribes to all topics available using Connext DDS' discovery service, and saves every sample to file (organised by topic) for future analysis (Fig. 5).

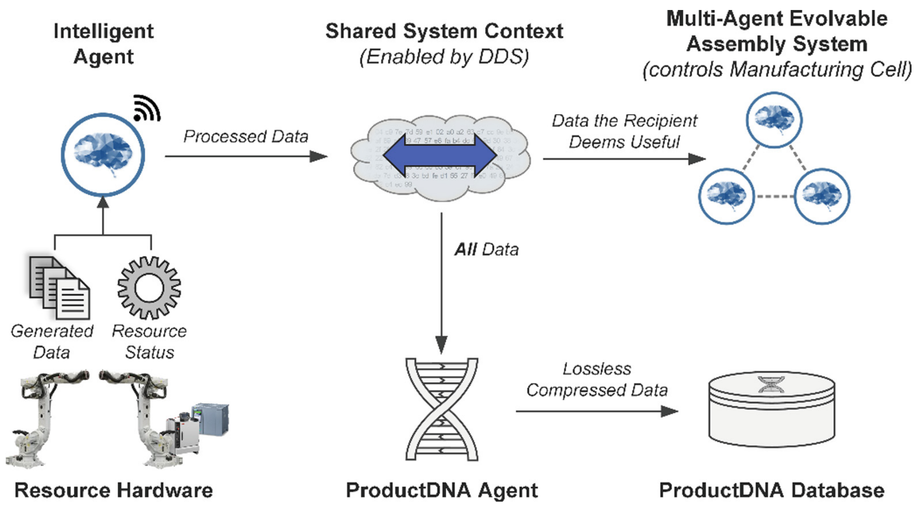


Fig. 5. Use of data in the solution. The agent (on the embedded computer) is able to retrieve information about the resource provided that information is presented on the resource user interface. This is shared (with optional pre-processing) to the shared system context, where other agents in the system can select what information they require to make optimum decisions. All data is stored by the ProductDNA agent.

4.4 Agents

Each resource in the system is controlled by an intelligent agent running on the embedded computer or smart device. We programmed agents in Java using the Java Agent Development Framework (JADE) [24, 25] which is an open source and flexible agent framework. On top of this, a BDI framework [15] was developed by which the agents reason about the demonstration cell.

The use of BDI agents has several advantages in this context. Firstly, there are strong synergies between the concept of beliefs and the use of a belief set in the BDI model, and of the use of data-centric communication, as the belief set of an agent is a subset of the shared system context representing the entire system, and the beliefs can be updated as the shared system context changes. Secondly, the BDI model decouples the storage of beliefs and the development of plans, from the execution of plans. As each agent may have a unique interface with its resource (or it may be a user interface in the case of a human worker), this simplifies the insertion of bespoke code whilst the rest of the code base remains homogenous.

The method by which the agents analyse recipes and determine how to execute capabilities to create a unique batch-size-of-one product is detailed in [16]. Users are able to submit recipes to the production line via an interface also running on the Dell laptop, which submits the recipe to the shared system context where the agents will check manufacturability and assign tasks.

5 Conclusions

This paper has presented a technical method for implementing Industry 4.0 and Evolvable Assembly Systems concepts on pre-existing legacy manufacturing cells, utilising data-centric communication, and intelligent agents executing on embedded computers which interface with their respective manufacturing resources. This method allows for a cost-effective manner in which smart manufacturing could be achieved by SMEs looking to include their production cells in the fourth industrial revolution.

One limitation of this approach is in not solving the issue of fixed actions. The use of capability topologies [10] helps mitigate the restriction of fixed actions, though does not solve the problem. The issue is inherent to the design of manufacturing cells, and in particular those not designed with maximum flexibility in mind.

Moving forwards, we aim to implement the user interfaces for human workers on smart devices such as tablet computers or smart watches, to make receiving and auctioning instructions, as well as relaying information back to the shared system context simpler. The entire demonstration cell will be integrated with the wider FA3D cell, during which the implementation will be extended to include the entire manufacturing cell, including challenges relating to interfacing with the more complex control system of the FA3D. We will also use additional models of embedded computer, including the Siemens IOT2000 [18] to experiment with system heterogeneity.

Acknowledgements. The reported research has been funded by the EPSRC grant EP/K018205/1, the support of which is gratefully acknowledged. We would also like to thank RTI for providing a license for Connext DDS Professional as part of their University Program.

References

1. Rhodes, C.: Manufacturing: statistics and policy, p. 13. House of Commons Library (2014)
2. Kagermann, H., et al.: Recommendations for implementing the strategic initiative INDUSTRIE 4.0: securing the future of German manufacturing industry; final report of the industrie 4.0 working group. Forschungsunion (2013)
3. Adolphs, P., et al.: Reference architecture model industrie 4.0 (rami4.0). VDI/VDE Society Measurement and Automatic Control (GMA) (2015)
4. Lin, S-W., et al.: The industrial internet of things volume G1: reference architecture. Industrial Internet Consortium (2017)
5. International Electrotechnical Commission: IEC 61131-3:2013 in programmable controllers - part 3: programming languages (2013)
6. Sommer, L.: Industrial revolution-industry 4.0: are German manufacturing SMEs the first victims of this revolution? J. Ind. Eng. Manag. **8**(5), 1512 (2015)
7. Jennings, N.R.: On agent-based software engineering. Artif. Intell. **117**(2), 277–296 (2000)
8. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT press, Cambridge (1999)
9. Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)
10. Van Brussel, H., et al.: Reference architecture for holonic manufacturing systems: PROSA. Comput. Ind. **37**(3), 255–274 (1998)
11. Barbosa, J., et al.: Dynamic self-organization in holonic multi-agent manufacturing systems: the ADACOR evolution. Comput. Ind. **66**, 99–111 (2015)
12. Chaplin, J., et al.: Evolvable assembly systems: a distributed architecture for intelligent manufacturing. IFAC-PapersOnLine **48**(3), 2065–2070 (2015)
13. Object Management Group Inc.: Data Distribution Service (DDS) Version 1.4 (2015). http://www.omg.org/spec/DDS/1.4/. Accessed 12 Nov 2017
14. Faccin, J., Nunes, I.: Modelling and reasoning about remediation actions in BDI agents. In: 2017 Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems (2017)
15. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: ICMAS (1995)
16. de Silva, L., et al.: Realisability of production recipes. In: 22nd European Conference in Artificial Intelligence (ECAI 2016), The Hague, Netherlands (2016)
17. Raspberry Pi Foundation: Raspberry Pi Zero W (2017). https://www.raspberrypi.org/products/raspberry-pi-zero-w/. Accessed 6 Dec 2017
18. Siemens AG: The intelligent gateway for industrial IoT solutions: SIMATIC IOT2000 (2017). http://w3.siemens.com/mcms/pc-based-automation/en/industrial-iot/pages/default.aspx. Accessed 6 Dec 2017
19. Raspberry Pi Foundation: Raspberry Pi 3 Model B (2017). https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed 7 Dec 2017
20. OPC Foundation: Unified Architecture (2016). https://opcfoundation.org/about/opc-technologies/opc-ua/. Accessed 5 Dec 2017

21. Siemens AG: SIMATIC WinCC V7 - maximum plant transparency and productivity (2017). http://w3.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/simatic-wincc/pages/default.aspx. Accessed 7 Dec 2017
22. S7Connector Members: S7Connector: Java library for S7 PLCs (2016). https://s7connector.github.io/s7connector/. Accessed 7 Dec 2017
23. Real-Time Innovations: RTI connext DDS professional (2017). https://www.rti.com/products/dds. Accessed 6 Dec 2017
24. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: Jade — a Java agent development framework. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming. MSASSO, vol. 15, pp. 125–147. Springer, Boston, MA (2005). https://doi.org/10.1007/0-387-26350-0_5
25. Bellifemine, F., Poggi, A., Rimassa, G.: JADE–A FIPA-compliant agent framework. In: Proceedings of PAAM, London (1999)