DOI 10.31256/UKRAS19.27

Development of a Simulated Production Environment for Plug-And-Produce Architecture Testing

William Eaton

***Abstract*— This paper describes the development of a simulated production demonstrator, used in the development of the openMOS plug-and-produce architecture. Primarily designed to allow synthetic testing of plug-and-produce technologies, the intention is to simulate real production hardware, such that there is no perceivable difference to the production controller. Communication with the main production controller is achieved via network, using individual embedded computers to act as PLC based 'device adaptors'. Each production device is also either simulated using the same embedded computer, or externally on a more powerful computer, with simulation specific information (such as material flow) transferred using ROS. Testing has proven the concept to work well, allowing for a larger demonstration of the openMOS project but at a fraction of the cost.**

I. Introduction

The global manufacturing industry is currently moving to the next generation of automation, often termed 'Industry 4.0'. One goal of this effort is to increase efficiency, allowing more products to be created to help satisfy the increasing global demand for consumer goods. At the same time, smart-manufacturing can also be used outside of traditional mass production, offering products in smaller numbers but with other advantages; such as reduced time-to-market, increased product complexity or lower production cost.

As automation equipment becomes more prevalent, Small and Medium-sized Enterprises (SMEs) are now rapidly investing in robotics and equipment previously only available to the largest and most technology advanced of companies. Although much of the technology is the same, SMEs face different challenges to large companies, typically due to costs. Unlike large companies which can afford bespoke solutions, SMEs are often limited to using existing "Off-the-shelf" equipment, sourced from whichever supplier is most cost effective. Production systems are then created by integrating equipment from multiple suppliers, often including 'legacy' systems with very low levels of automation.

To simplify this process and make the benefits of smart manufacturing more obtainable, efforts are being made to develop unifying "smart-automation" architecture, capable of integrating varied equipment into a cohesive system. This form of architecture must combine both software and hardware elements, to communicate and control the diverse range of equipment used in manufacturing. The work described in this paper has been undertaken during the development of such a system, as part of the openMOS (**Open**-source **M**anufacturing **O**perating **S**ystem) project [1].

*Research supported by the European Commission.

W.H. Eaton is with the Intelligent Automation Centre, at Loughborough University, UK. (e-mail: w.h.eaton@lboro.ac.uk).

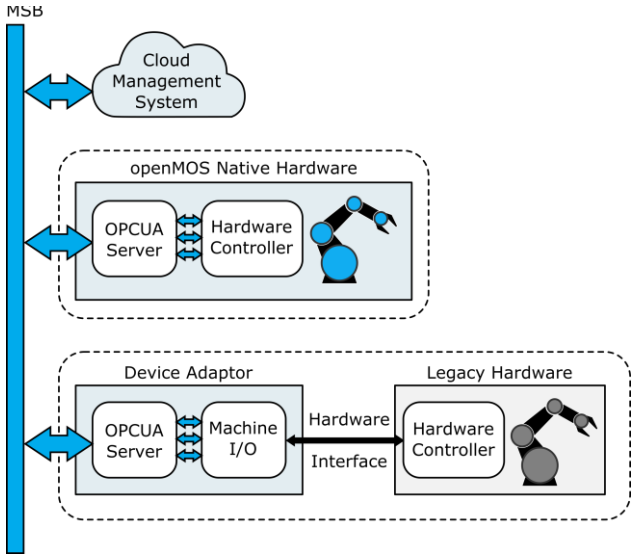

Figure 1. High-level openMOS Architecture for native and legacy equipment

A. *openMOS Architecture*

The openMOS project is an openly-accessible software-architecture, intended to provide a standardised platform for automation in manufacturing. One of the key goals of the project is to provide 'plug-and-produce' (P&P) [2] functionality, allowing any compatible equipment to be integrated into a system with minimal manual setup. Although other P&P systems have been proposed, they typically focus on low level integration, such as local-discovery, where each device announces itself upon connection [3] [4].

Although such functionality is required, these systems do not currently attempt to solve the more difficult task of conveying to the rest of the system what the new device can accomplish. For openMOS, this is achieved by abstracting the *product* from the equipment as much as possible. For a device to be compatible, its capabilities must be encapsulated as 'skills'. (These could be entirely unique or highly generic skills such as 'hole drilling'.) To provide P&P functionality, each device declares its skills and parameters to the overall system upon initial connection. Each product is then defined using a 'recipe', which is an ordered list of skills which must be applied to the input material for the final product to be created. As transportation between equipment is also considered, the most efficient flow of material through the manufacturing facility is automatically determined. These features are intended to facilitate 'Flexible automation', [5] in which either existing equipment can be rapidly reconfigured to meet changing demand, or new equipment added to increase capability.

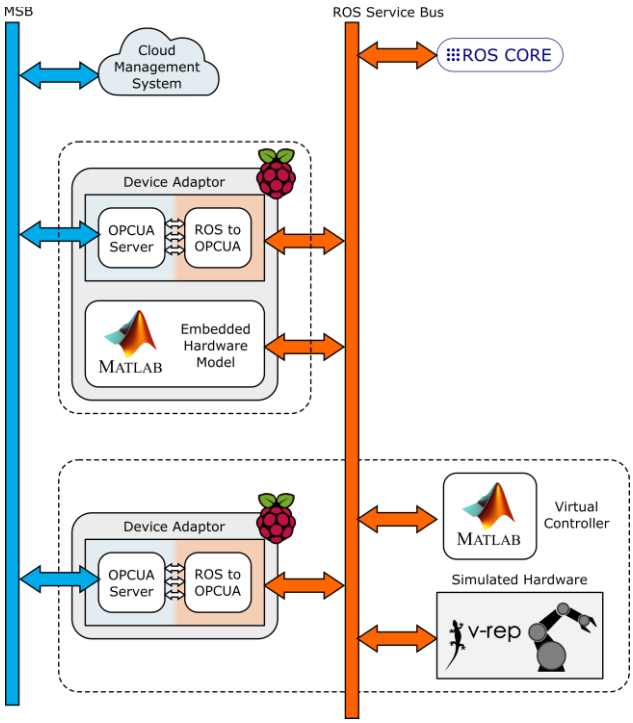

Figure 2. High-level openMOS Architecture for simulated equipment

During production, the core functionality requires all devices to be connected to a common Manufacturing Service Bus (MSB) [6] running on a local network, as shown in Figure 1. The MSB is used to transmit "skill" requests and sensor data, providing every device access to any data it requires. Overall control of the production task is achieved via an agent-based Cloud Management System (CMS). For a device to be openMOS compatible, it must fulfil both hardware and software requirements. Although new production devices featuring openMOS are natively compatible, existing devices are usually not equipped to support P&P interfaces and may lack network connectivity, making interfacing difficult. Therefore, legacy support is introduced through 'device adaptors' which interface between the device and the MSB. Each DA is essentially a small embedded computer, which converts openMOS skill commands into an equipment specific interface.

B. *Simulation*

For any automated manufacturing environment, the cost of equipment represents an enormous initial investment. As the openMOS architecture has been co-developed by several SME's and academic institutions, the scale of testing has been limited. Therefore, although several physical production demonstrators have been created, each has been a compact representation of a specific capability of the overall architecture. Instead, it was decided that a simulated approach could be used to demonstrate openMOS capabilities at a larger scale. Furthermore, the development of a simulation approach would be useful for any SME wishing to adopt the openMOS platform, allowing them to verify the functionality of the system without applying it to their production environment. This paper presents the development of a scalable approach to virtual system development, intended to provide multiple levels of fidelity.

II. Simulator Requirements

A. *Computational Hardware*

Although the term 'simulator' is used here to describe the entire system, it should be recognised that there are two distinct parts; the first being the openMOS architecture itself. For the results to be meaningful, openMOS must be deployed using hardware comparable to what would be used in a production environment. This applies to both the computers which run the software, as well as the networking equipment and interfaces for each machine.

The second part is the simulated plant operations used to produce representative data. At the most basic, this consists of responding to openMOS skill requests and replicating the sensor data that would be created by actual equipment. As a single production facility could include many hundreds of connected devices, the computational requirements could be quite large, especially for high fidelity simulation. Therefore, as the primary purpose of the simulation is to test the performance of the openMOS control system, it is essential that the simulator must be implemented in a way does not affect the performance of the MSB. This is most easily achieved through hardware separation, in which openMOS and the simulator are run on independent equipment. Moreover, as using a single computer limits the scale of the manufacturing process which can be modelled, a distributed computing approach is to be used; with multiple computers working in conjunction to produce a facility scale simulation.

B. *Distributed Computing, Networking and Device Adaptors*

To ensure that the demonstrator is representative, the interface between openMOS and the simulation must replicate what would be found on an actual deployment. One critical element is that the openMOS communication protocols rely on each device having a unique IP address, to ensure that each device can be separately addressed by the controller. Returning to Figure 1, there are two methods for openMOS to interact with hardware:

- Natively supporting openMOS within the hardware controller itself
- Using a dedicated openMOS device adaptor to support legacy equipment.

Native support of openMOS is the eventual goal of the project, with manufacturers producing devices already compatible with openMOS installations. However, as the devices within this work are simulated, such an approach would require including openMOS interface protocols directly within the simulation. The downside of this approach is that any latency in the simulation could introduce delays to the MSB which are actually specific to the simulator, rather than openMOS itself. Furthermore, although it is possible that both network interface and controller could be virtualized, to ensure compatibility it is far simpler to use separate physical devices. This also allows the 'Plug and Produce' functionality to be tested directly, simply by unplugging equipment to represent failure, and attaching new equipment during production to test how the system responds. As such, custom hardware device adaptors will be used to communicate between the simulation and openMOS.

III. SCENARIO

Although the simulation approach has been developed to allow for a large number of devices, the initial implementation is more modest, aiming to begin with twenty simulated devices and their device adaptor counterparts. The main goal of this initial setup is to demonstrate flexible process simulation, in which machines can be added or removed from the production environment.

The chosen scenario is the assembly of a generic electronic product consisting of four components: two casing parts and two electronic internals. Both the product casing and internals have variants allowing for individual products to be customised. The simulated facility consists of several different workstation types (each with several instances) through which the products flow in a non-linear order, as shown in Figure 3:

- 4 Laser cutting stations
- 3 Painting Stations
- 3 Gluing Stations
- 1 Final Marking Station
- 2 Assembly Stations
- 2 3D-Printing Stations
- 6 Assembly Stations

Variation in each individual product can require parts to revisit workstations multiple times, requiring the CMS to determine the best use of available equipment. For example although there are 3 distinct assembly tasks, the 6 assembly stations are each capable of performing every action, and can adaptively share workload. Finally, product flow through the simulated facility is achieved using a small number of Automated Ground Vehicles (AGVs). A single AGV controller is used to assign jobs to AGVs, which move products on predefined routes.

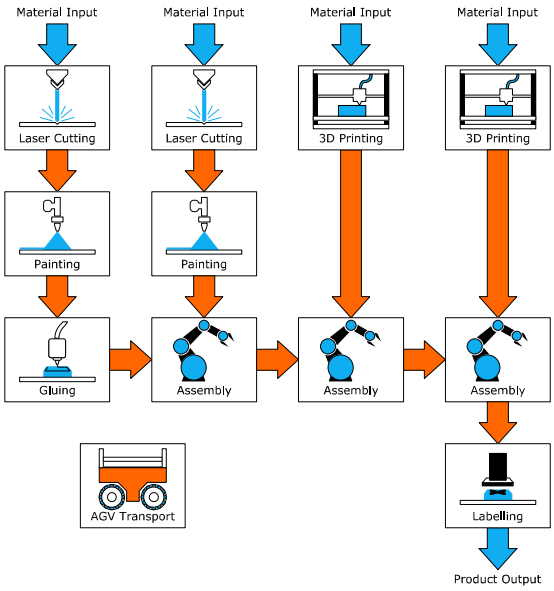

Figure 3. Material flow between stations for single product completion within simulation scenario.

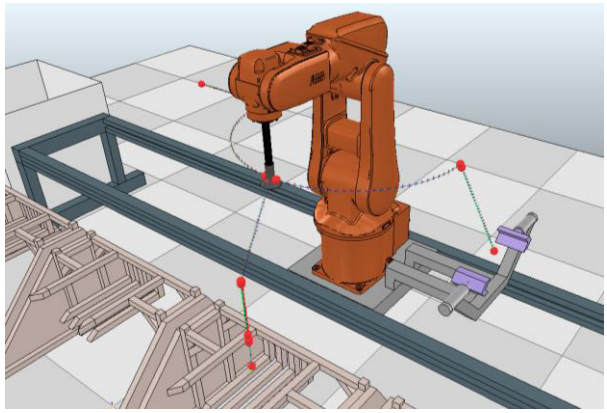

Figure 4. A virtual assembly workstation shown in VREP, along with the preplanned motion paths.

IV. IMPLEMENTATION

To achieve the requirements of the simulation, several different software packages were combined across a range of different hardware. Figure 1 shows a high-level depiction of the software setup for openMOS, as it would be deployed in an actual production environment. Figure 2 shows a similar depiction of the software used in the simulation. Comparing Figures 1 and 2, the most obvious difference is the inclusion of the Robotic Operating System (ROS) to act as a secondary communication protocol. As the simulation is distributed, consisting of equipment controllers and simulations, ROS was identified as the most suitable method of communication within the simulation itself, without interfering with the openMOS MSB. With its wide adoption and ease of use, ROS also allows an extensive range of existing software to be easily deployed, or additional capabilities to be added quickly using its well-defined software architecture. As previously discussed, for each simulated device within the demonstrator, an embedded computer was required to act as the device adaptor. Owing to their popularity and wide range of software available, the decision was made to use the Raspberry Pi 3 (RasPi) in this role.

In an actual openMOS installation, communication between devices is primarily achieved via an Open Platform Communications Unified Architecture (OPCUA) [7] server/client model. Each device hosts an OPCUA server (which is essentially just a database of variables which can be accessed or updated in real-time), allowing connected clients to read/write data as appropriate, via the MSB. Due to the success of OPCUA in communicating between the device adaptor and the MSB, OPCUA was also seen as an appropriate method for the device adaptor to communicate with the simulation (effectively running two OPCUA servers for each device adaptor). Therefore at minimum, each RasPi is running two OPCUA servers – one 'facing' the MSB and one 'facing' the simulation – and acts an 'adaptor', converting openMOS skill requests into ROS based simulation triggers.

For the OPCUA server which connects to the MSB, a standalone Java OPCUA implementation was used, to maintain consistency with how a typical openMOS install would function. By contrast, the simulation facing OPCUA server was implemented using the ROS_OPCUA package,

which directly maps ROS topics and services into a OPCUA database. This simplifies integration to the extent that when the device adaptor identifies an openMOS skill request on the MSB-facing OPCUA server, it can simply set a variable on the simulation-facing OPCUA server to trigger the appropriate ROS service within the simulation.

As shown in Figure 2, for extremely simple simulated devices, (such as the 3D printing stations which simply output a product at a fixed rate) the required simulation was simple enough to run on the same RasPi. This was achieved using an Embedded Simulink model, which communicated with the OPCUA server onboard via an additional ROS node on the same RasPi. For all other stations, a higher fidelity simulation was used to model the physical interactions of equipment and product. This allowed the quality of the output (such as paint coverage and glue distribution) to be assessed, to confirm that the virtual station models were appropriately complex as to represent real equipment. For these stations, Matlab was used as a virtual controller, with the simulation provided using the VREP simulation environment on a dedicated desktop PC per station. An additional standard desktop PC was connected to the openMOS network to run the CMS and MSB software.

Finally, to provide high speed network connections between all devices, two commercial DLink DGS-1100-24 Gigabit Switches were used to provide network connectivity typical of a small production environment. Although only 20 devices were used in this initial scenario, 48 RasPi device adaptors were created for future simulation scenarios.

V. Assessment of Suitability

As stated in the introduction, the purpose of this simulation is not to produce the most accurate reconstruction of a production environment, but is instead intended to provide a facsimile of the interactions between openMOS and a production system, without having to invest in the hardware. The simplest method of assessing whether this solution is suitable is to compare a simulated workstation with a practical example; as the simulation is intended to replicate the data provided by equipment, the CMS should not be able to differentiate between simulated and practical equipment.

For this purpose, one of the simulated stations within the scenario was selected to exactly mimic an actual workstation already available. The virtual `gluing station' was designed to replicate the motions and actions of a physical ABB IRB120 industrial robot arm. In addition to the robot itself, a gluing nozzle and temperature sensor were also included. By creating a ROS package for the physical system, the software implementation was nearly identical to the simulated devices, requiring minimal integration. As the ROS_OPCUA adaptor automatically exposes ROS topics and services, there was no additional setup beyond allocating separate ROS namespaces to each station so as to differentiate between them. (As such, it can be also be concluded that that integrating physical equipment into the simulation is extremely easy, provided that a ROS package is available).

Having installed both virtual and physical versions of the same workstation, both systems were linked to the MSB and an empirical analysis of the data carried out. As there was no substantial difference in the data produced, the simulation was judged to be an accurate representation of the actual robot, validating that the simulated approach produced an accurate portrayal of production hardware.

Following this, repeated testing has been undertaken to validate the P&P functionality of openMOS, by removing RasPis from the simulation during operation. It has been found that the system is capable of responding to this by re-routing product to additional instances of the same workstation, where possible. When these machines are then restored, openMOS is once again able to make use, without the need to restart the entire facility. As both simulated and real-world process devices can function together, the simulation can be assessed to produce an output equivalent to that provided by actual hardware. Therefore, the simulation is capable of generating network traffic representative of an actual physical production facility, but at a greatly reduced cost.

VI. Conclusion

This paper has outlined the creation of a simulated production system, designed to replicate the data and signals from real world equipment to verify many aspects of the openMOS architecture. The developed simulation approach is highly straightforward, in addition to being easily expandable to simulate production facilities of greater size.

Following on from the initial testing covered here, the intention is for future work to include a comparison between the simulation and a hardware infrastructure with known timings; intended to allow an objective comparison to determine if openMOS is truly capable of performing on a large scale without performance loss. Finally, the simulation infrastructure developed here also has the potential for training activities, as well as potentially serving as a basis for benchmarking and certification

References

[1] Openmos website. [online] Available at: https://www.openmos.eu/ [Accessed 05 Apr. 2018].

[2] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, J. Ota, “Agile Assembly System by “Plug and Produce”,” CIRP Annals, Volume 49, Issue 1,Pages 1-4,

[3] L. Durkop, J. Imtiaz,H. Trsek, L.Wisniewski, J. Jasperneite. “Using OPC-UA for the Autoconfiguration of Real-time Ethernet Systems”. In Proc. 11th IEEE International Conference on Industrial Informatics (INDIN), 2013, pp. 248-253.

[4] V , Hammerstingl., G. Reinhart. “Unified Plug&Produce architecture for automatic integration of field devices in industrial environments”. In Proc. IEEE International Conference on Industrial Technology (ICIT), 2015, pp. 1956-1963.

[5] P. Neves, L. Ribeiro, J. Dias-Ferreira, M. Onori and J. B. Oliveira, “Layout validation and re-configuration in Plug&Produce systems”, Journal of Assembly Automation, Volume 36, 0pp. 412-428, 2016

[6] N. Lohse, P. Ferreira, I. Pereira, “Deliverable: D3.1: Open Plug and Produce Architecture Specification”, [online] Available at: https://www.openmos.eu/ [Accessed 05 Apr. 2018].

[7] B. Lydon, “Non-Proprietary Controller-to-Controller Communications” February, 2014, [online] Available at: https://www.automation.com/portals/manufacturing-operationsmanagement/opc/non-proprietary-controller-to-controllercommunications [Accessed 05 Apr. 2018].