An event-based AutomationML model for the process execution of ‘Plug-and-Produce’ assembly systems

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Citation: DANNY, P. ... et al, 2018. An event-based AutomationML model for the process execution of ‘Plug-and-Produce’ assembly systems. Presented at the 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal, 18-20 July 2018, pp.49-54.

Additional Information:

- IEEE © 2018 Crown. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Metadata Record: https://dspace.lboro.ac.uk/2134/36357

Version: Accepted for publication

Publisher: IEEE

Please cite the published version.
An Event-Based AutomationML Model for the Process Execution of ‘Plug-and-Produce’ Assembly Systems

Paul Danny, Pedro Ferreira, Niels Lohse
Loughborough University, Wolfson School of Mechanical, Electrical and Manufacturing Engineering,
EPSRC Centre for Innovative Manufacturing in Intelligent Automation, Holywell Park,
Loughborough, UK – LE11 3QZ
[P.Danny, P.Ferreira, N.Lohse]@lboro.ac.uk

Abstract: Assembly systems today are facing significant pressure to deliver high performance process executions, while being responsive to the fluctuating market demands. However, the implementation the trending Cyber Physical Systems concepts via ‘Plug-and-Produce’ devices produces some communication overheads. In this direction, the openMOS project aims to decouple the elements that are responsible for adaptation and general operations of the system. This allows the system to have two parallel processes. Towards this end, the priority is to deliver high performance process executions, while the other process focuses on delivering the required agility. The focus of this work is narrowed down to the development of task execution tables that guarantees high performance process executions. In this direction, the definition of task execution table is based on an existing AutomationML (AML) model that highlights the explicit relationships between the Product, Process and Resource (PPR) domains. A new decisional attribute has been added to the existing ‘Skill’ concept, which provides the flexibility to incorporate event-based process alternatives. An insight description on how the system handles process executions during run-time failures is also provided. Finally, this paper illustrates the run-time implementation of the execution table with a help of an industrial case study that has been used for a demonstration activity within the openMOS project.

Keywords: AutomationML; Task Execution Table; ‘Plug-and-Produce’; Cyber Physical Systems; Semantic Models; Product; Skill; Skill Recipe; Skill Requirement

I. INTRODUCTION

The global smart manufacturing industries is expected to worth nearly 67 billion USD by 2020, with a compound annual growth rate of 6% from 2014 to 2020 [1]. In this landscape, there are already several country specific research consortiums and initiatives such as: Advanced Manufacturing (US), e-Factory (Japan), Indusrie 4.0 (Germany) and Intelligent Manufacturing (China) [1]. The manufacturing enterprises are also constantly aiming to take advantage of technological advancements to produce agile, reliable and lower cost systems.

The advent of technologies such as: ‘Plug-and-Produce’ and Internet-of-Things (IoT) are playing an important role in enabling a new manufacturing paradigm commonly known as Cyber Physical Systems (CPS) [2]. CPS encapsulates computational entities, communication networks, data accessing and data processing services, which are very closely connected to its physical environment and can rapidly respond to its process variables. This enables the production systems to adapt quickly towards the random real-time changes that are expected in any multifaceted production environment [3]. However, the current CPS capability demonstrations for smaller test scenarios reveals its underperformance in relation to the tradition control approaches. This is mainly due to larger requirements in terms of their decision-making processes, resulting in time delays in the execution process. This is crucial for industries dealing higher production volumes, as any compromise in the performance of the system can cause significant financial loss.

In this landscape, the vision of the openMOS project [4] is to enable high performance ‘Plug-and-Produce’ systems that supports rapid configuration, integration and reconfiguration without compromising their operational performance. The idea is to provide a common Manufacturing Service Bus (MSB) that can interface existing industrial control systems to higher level applications. Furthermore, the openMOS cloud environment provides a virtual copy of the physical system, which can be used to optimize the performance of the system. However, there should be efficient models and supporting mechanisms in place for the lower-level process executions.

Hence, the main motivation behind this work is to develop mechanisms that retains agility without compromising the performance of the system. Therefore, this paper reports on the development of the openMOS project’s process execution model that establishes the explicit relationship between the product and the process, while highlighting the features of operating the model.

II. LITERATURE REVIEW

The introduction and implementation of CPS in the existing factories, i.e. to migrate manufacturing systems towards distributed, interoperable and intelligent IOT technologies would require a paradigm shift [5] [6]. The implementation of such architectures needs smart ‘Plug-and-Produce’ devices for easy integration and management of device configurations and control setups. In simple terms, it is expected that when a device is connected to the network of other devices or on to the cloud, it should register and realize its topological position with their networked partners [1]. Furthermore, the newly added devices should also be able to communicate with other devices regarding their capabilities for process orchestration [7].

Therefore, the idea is to enable seamless integration of production equipment for dynamic information exchange, and a
holistic optimization of resource utilization. The reference architecture provided by Industrie 4.0 [8] mentions OPC Unified Architecture (OPC UA) [9] for the facilitation of semantic description, information modelling and online registration of manufacturing entities. Recent research activities include the description of Product, Process and Resource (PPR) domains to facilitate the automated generation of process sequences and system configurations for production requests [10]. This increases the need of the manufacturing equipment to be more self-aware, agile and responsive towards unplanned breakdowns and rapid fluctuations in production requests [1].

In any case, today’s production setups are controlled by several layers of automation pyramid, such as: PLC, MES and ERP systems. PLCs are reliable, fully deterministic and very responsive for real-time applications. Over the years, PLCs have evolved significantly to cope with the introduction of new standards and open architectures for the distributed control of production systems. IEC 61499 is one of the standard that allows provides an object-oriented function block that allows the reuse of codes for faster control and integration for distributed systems [7]. Also, [11] proposed the advantages of using agent-based approaches for controlling distributed production setups. The main advantage of using agent-based systems lies in its distributed nature, where the problems are solved through various communication strategies. Nevertheless, the application of the agent technology in real-time industrial applications is still not favored. This is mainly due to the extra overheads produced by the communication and negotiations between the agents.

Considering this [12] proposed a hybrid approach to combine the relative strengths of PLCs and agent-based systems. This approach focuses on retaining the performance of the PLCs, while adding the adaptability and reusability of software elements with agent-based systems. Moreover, it allows effective decoupling between the agent communications and PLC-based process executions. However, the performance of the agent-based approach towards complex decision would always imply time delays in production. In the following, PLC suppliers have also realized that enabling interfaces for higher level programming languages can lead to the development of open control approaches [12].

Similarly, the existence of predefined and harmonized standards is critical to achieve high interoperability and rapid integration. IEC 62714 is one of the most promising solution for data exchange focused towards the domain of automation engineering. ‘AutomationML’ (AML) is a data format defined with in IEC 62714, which is XML schema based and has been developed to support dynamic data exchange in any heterogeneous engineering environment [13]. The main aim of AML is to join engineering tools from different domains such as: process control engineering, mechanical plant engineering, electrical design, process engineering, robot programming, PLC programming, HMI development etc. In the following, [14] proposed an AML model that makes use of existing AML concepts to formalize the use of ‘Plug-and-Produce’ components, while enabling the next generation of CPS. This is achieved by the definition of new libraries within AML to support the definitions such as: ‘Skill Requirements’, ‘Skills’ and ‘Skill Recipes’.

The definition of assembly process is crucial as this can have a significant impact on the performance of the system. This is simply because, it bridges the gap between the product that needs to be assembled and the manufacturing system that assembles it. Also, the definition of assembly processes are derived from the product’s assembly requirements (‘Skill requirements’). The functional capability of the equipment modules are defined as ‘skills’. The instructions for the execution of an equipment’s ‘skill’ is formalized as ‘skill recipes’. The ‘Skill Recipes’ are purposely generated to fulfill the product’s ‘Skill Requirements’. Furthermore, the product definition in AML establishes the links between all the possible process alternatives (‘skill Recipes’) for the current setup of the physical system. Hence, these three definitions provide the necessary relationships between the ‘Product’, ‘Process’ and ‘Resource’ (PPR) domains.

However, the previously proposed AML model did not consider the process executions but focused more on establishing the relationships between the PPR elements of the model. In this sense, this paper focuses on using the existing AML model [14] that establishes the relationships between the PPR domains to define a new process execution process that can be used for enabling high performance production and enable fast system adaptations.

III. PROBLEM DEFINITION

The current CPS developments provide the deployment of ‘Plug-and-Produce’ in today’s industries, which is expected to deliver system agility and responsiveness to market changes. However, current solutions tend to underperform during execution in relation to conventional control approaches. This is expected as these systems require several additional communications with its networked devices, as well as considerations at each device to make decisions. Consequently, this increases the latency in the system, resulting in its underperformance in relation to the dedicated control approaches. In small systems, where this CPS has been demonstrated, these issues are not significant. However, in large and complex systems, this is expected to have major implications. Nevertheless, the capabilities that deliver agility and quick adaptations are a clear market need which cannot be achieved with conventional control approaches.

The current architectures for CPS provide solutions that target constant system adaptations or new product introductions. However, the need for such systems is still not there, as the products tend to be defined in a relatively stable and predictable manner. The merits of CPS can be fully realized when all the products entering the production line are unique, as the system would need to be constantly adjusting to the changes. However, the shift from mass production to mass customization is an ongoing process in industry, thus we are still far from every product being unique. Instead, the number of product variants and the production volume may vary according to market demands. The customization and system adaption to market demands is required, but not a constantly adapting system. Therefore, there is a possibility to take advantage of this relative predictability and establish a system that operates not expecting constant change, but rather able to deal with changes when they arise. This means that the focus on the system operation can be
on its performance, while still enabling it to adapt when needed. This would deliver a system that does not respond immediately to changes, but that does so as soon as it can, as to not disrupt its operation. The idea is not to limit the agility of the system, but to focus on delivering higher performance systems.

The criticality of this approach lies on the mechanism that supports the interactions between the two processes. In this sense, CPS approaches already deliver a direct and constantly updated link between the physical device and its virtualization. This means that the mechanisms to detect if a device is unplugged from the system are already in place, which means that all elements in the systems could understand the impact of this removal. If a device was expected to execute an assembly process, the system would not need to find an alternative. In current CPS, this would not be a problem as this would just use the usual process, however because the proposed approach does not react immediately the system execution would have to stop until a decision is made on the new production settings. This process can take a bit of time, particularly in very complex systems, therefore it would be important for the system to be resilient and find a quick alternative that is able to keep the system running, even if in sub-optimal conditions. Therefore, the system execution process would also need to consider a mechanism for this to be achieved.

The openMOS project builds on CPS solutions and establish a Manufacturing Service Bus (MSB), which provides the means to aggregate all elements in the CPS. This means it is responsible to deliver the necessary interfaces that links the cloud environment and the physical system, while providing translation in the data formats when needed. This enables frequent event-based updates to the cloud regarding the current state of the physical system. The cloud in the openMOS project is expected to deliver on the system optimization and adjustments based on changes as shown in Figure 1. This means that there needs to be mechanisms in the MSB to propagate these changes as it is responsible for all communications in the environment.

The MSB is also responsible to provide interfaces with other tools, including tools that introduce new products and new product orders. These need to be disseminated to the several elements in the environment, but most importantly there needs to be a means to start the production of products in these orders. Since the MSB sits in between the two decoupled processes, it is the logical choice to deliver orchestration between product orders and triggering the production execution. This means that the MSB needs to be able to understand the models and concepts of the data it is handling. Current semantic model provides the relationships between process, process and equipment, while not really addressing the execution process. To deliver a solution for this one needs to establish the process execution model in the wider CPS models, which can be used to drive the execution process which will be orchestrated by the MSB. This paper focuses on using an existing AML model [14] that provides all the PPR relationships, and incorporate a solution that supports high performance process executions through clear decoupling of system operation and optimization. The details regarding the structure and the contents of this solution will be discussed in the next section.

In order to achieve this, the first step is to decouple the elements responsible for system operation and its adaptation. This means that the elements related to system execution should be independent of the adaptation elements. However, to enable CPS one needs to establish a one to one relationship between the physical and the virtual entities. This will allow the virtual cloud domain to analyze, optimize and forecast the best settings to increase the performance of the system. On the other hand, the physical devices will continue to operate with an existing setting, until the cloud provides an update, or a disruption occurs. This means that the overall functionality of the system will be achieved through these two parallel processes, which can have separate computational resources. The first focuses on the delivering high-performance process executions by minimizing the communication and decision-making processes. The second focuses on the optimization and adaptation of the system in response to any relevant aspects, such as new product introductions, adjustment of volumes, failures in machines, etc.

The primary objective of this model is to deliver CPS with a more comparable performance to current control solutions. The vision is to deliver production systems that adapt to deal with varying production demands, without compromising its operational performance. In this direction, AML has been identified as the prime candidate for modelling, as it allows the integration of current legacy systems with other tools [14]. AML enables the self-description of ‘Plug-and-Produce’ devices, which can include both physical and functional information. This enables the virtualization of the entire physical system in the cloud. This is crucial for CPS, as it requires an exact virtualization of the actual physical and operational information to deliver its vision.

Another main motivation towards the use of AML is that it provides the detailed relationships between the PPR domains, which is crucial to this work. Towards this end, an existing AML model [14] has been used to establish these PPR relationships.

The definition of assembly product in AML provides the
relationships between all the available ‘skill recipes’ that can be used to fulfill its ‘skill requirements’. This means that there can be more than one ‘skill recipe’ that can fulfill a ‘skill requirement’. The skill recipes are purposely generated with one or more parametric settings for the execution of skills to fulfill the product’s skill requirements. Therefore, the definition of skill recipe establishes the relationship between the skill requirement that it fulfills, and the equipment module that executes it.

In the openMOS project, all the equipment modules are ‘Plug-and-Produce’ capable and are networked together via the common MSB. The MSB establishes the necessary interfaces for the equipment modules to communicate regarding their process capabilities and other information related to their operation and maintenance. Also, the device adaptors developed within the project serve as service wrapper that enables all the lower-level controls within the device. Also, the MSB’s cloud interface provides updates about the physical system and its capabilities at any given time. This implies that the information is constantly available for optimizing the system. The current CPS systems are designed to establish these optimizations during runtime. However, this implies that the system is in constant calculation, and subsequently leads to significant time overheads.

To overcome these overheads, it is necessary to decouple the elements responsible for system operation and its constant optimization. This means that the executions in the physical system should be independent of the higher-level system optimization in the cloud. Accordingly, the system optimization component will determine and define the best settings for the system to produce the current product. This implies also the ability to replace the current settings for process execution in the system. It is also important to note that the system optimization can deploy changes to the system to deal with new product introductions, adjustment of volumes, equipment failures, etc. On the other hand, the production system can simply execute the solution provided by optimization element. Of course, given the dynamic nature of CPS solutions might become unavailable. This can be mainly due to disruptions in the system, such as equipment failures or the optimization element decided to update the current solution. It is critical for the system to maintain some resilience to deal with disruptions, and should be able to rapidly adapt without halting the production processes. While only one setting should be active, the existing AML model’s PPR relationships can be used to determine a quick alternative to continue the production processes. This would deliver some leeway in time, so that the system optimization can recalculate and provide a new set of settings.

Another important aspect is that the openMOS product definition allows the user to set production orders that can either include generic products, or specific product instances. This feature provides the flexibility to define a whole new order of unique products, or a few product variants. Consequently, the process execution mechanism should deal with both generic products and product instances. Therefore, the solution needs to incorporate flexibility for the system to execute generic products or product instances. If the instructions for a product instance already exists, this should be executed by the system, otherwise the system by default will execute the instructions for the generic product. In case, if the system lacks one or more matching skill recipes for a product, then the MSB will highlight that there is not enough information to create the specified product order.

This work proposes the creation of a table, which provides quick lookup capabilities for fast system response, that will capture the settings of the system. This means that it will establish the relationships between the product and the active recipe for its execution. Figure 2 graphically presents the semantic structure and the contents of this table (task execution table). Accordingly, a table can include one or more task execution lines, and each of those lines will include four main

Figure 2: Overview of the Process Execution Model

Figure 3: Overview of the openMOS Framework
attributes. The first attribute refers to a product id, which could either be an instance of a product or the id of a product variant (generic product). The second attribute refers to a skill recipe id that belongs to an equipment in that sub-system. The pairing of the product and recipes provide the means for the device to identify what will be executed. The third attribute refers to the id of the subsequent skill recipe that should executed. It is important to note that this skill recipe may belong to the same or a different sub-system. This means that the information on the next steps to every product needs to disseminate to the appropriate devices. While this makes the optimization process more complex, it provides faster response on the execution side. The fourth attribute represents a list of possible skill recipes that can be used to overwrite the next recipe to be executed. This can have entries only when the ‘Skill’ being executed has a decisional attribute, which implies this capability.

Figure 3 depicts the expected behavior of the system in steps as follows; Step 1: the product is deployed in the MSB; step 2: the process orchestrator who has the knowledge of the execution tables communicates to the device adaptor to execute a specific task execution line; step 3: the adaptor reads the line and executes the current recipe, and informs the process orchestrator about the subsequent execution; step 4: the process orchestrator receives this information and finds the corresponding task execution line, and instructs the device adaptor to execute it. Also, it is important to note that the subsequent recipe execution may belong to the same or a different equipment). This process continues unless and until the information in the execution tables are valid. Disruptions such as equipment failures, or unplugging a module from the system may result in a set of invalid execution tables. Subsequently, the system optimization element in the cloud analyses the change and updates the execution tables to resume system operations.

V. ILLUSTRATIVE EXAMPLE

The illustrative example is based on the openMOS project’s industrial demonstration case, which presents a closed loop production line as shown in Figure 4. The setup consists of several workstations that are connected to one another by means of transport systems. Each of these transport systems include one or more conveyors and junction modules. The conveyors are used for linear transport and junctions for selecting one of the several paths offered by the conveyors ahead. This is a purpose-built setup to inspect the product for leaks and follow appropriate corrective measures if the product fails the test.

Accordingly, the assembly product has been defined with a list of skill requirements that follows a specific sequence (precedence) to complete the assembly process. The addition of the new decisional attribute allows the product requirements to accommodate optional sequences. Figure 5 highlights the optional sequences for the product’s skill requirements in two different colors. Here, the leak testing skill includes a decisional attribute, which produces an integer output after its execution. This value is crucial for the device adaptor to decide if it should overwrite the appropriate subsequent recipe to execute.

In this illustrative example, the first step is to launch the product AML into the MSB with its respective order. Figure 5 shows that the product AML includes all the PPR relationships to highlight all the available skill recipes for current system configuration. The second step is initiated by the process orchestrator by pointing the task execution line that can fulfill the product’s first skill requirement. Accordingly, the device adaptor of the manual loading station receives the instruction to execute this specific line. The third step will be initiated by the manual loading station’s device adaptor by sending the instruction for the subsequent recipe execution. In this sequence, once the leak testing skill is executed, the device adaptor can overwrite the subsequent process based on the output value of the decisional attribute as shown in Figure 5. It is also important to note that the cloud environment receives constant updates regarding the ongoing process executions in the system. Consequently, it can optimize the system based on the information that it receives from the normal operation.

In case of any disruption, or if a device is unplugged from the system, then all its corresponding skill recipes will become unavailable. For instance, if the leak testing station1 is unplugged from the system, then the MSB will get this update through the missing “equipment heart beat”. Subsequently, the MSB will check the product AML, if any alternative skill recipe exists for the current skill requirement. In this case an alternative exists and it belongs to the leak testing station2. Then, the process orchestrator communicates with the all corresponding device adaptors to route the product to leak testing station2. This allows the system to react quickly without the need to wait for the system optimization or stopping the production process. It is also important to note that the optimization element in the cloud can even add new lines to the execution tables. This allows to change the behavior of system to its increase its operational performance.

VI. CONCLUSION

This paper presents a process execution model which enhances the AML model towards the high-performance process executions in ‘Plug-and-Produce’ assembly systems. The process execution table is modeled in such a way that it supports high performance process executions without compromising the adaptability of the system. The proposed solution allows the openMOS system to respond quickly for minor changes in the
It is important to note that the adaptability offered by the execution tables does not guarantee an optimal condition for the system to operate. Instead, the system continues to operate without the need to stop or wait for decision support. However, the system optimization element in the cloud, continuously monitors the physical changes in the system and calculates the best setting for the system to operate. These settings are nothing but the execution tables, which will be constantly updated by the cloud environment to improve the performance of the system.

The illustrative example from an industrial demonstration case highlights the interactions between the MSB and the device adaptor during run-time. The preliminary test results show that the performance of this approach is very close to the existing systems. Future work will be focused on validating this approach by providing clear metrics for comparing this approach with the existing industrial control systems. The test results from this comparison will establish the benchmarks for future reference.

**ACKNOWLEDGMENT**

The reported work is a part of openMOS project partially funded by the European Commission as part of EC-H2020-IA (GA 680735). The support is gratefully acknowledged.

**REFERENCES**


