An AutomationML Model for 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

Magno Guedes
INTROSYS – Integration for Robotic Systems SA, R&D Department, Estrada dos 4, Castelos Lote 67, 2950-805 Quinta do Anjo, PT
[Magno.Guedes]@introsys.eu

Abstract: ***This paper aims to support the creation of high performance 'Plug-and-Produce' systems by proposing a new semantic model that targets the use of AutomationML (AML). In this direction, the focus is narrowed to the self-description of equipment modules that highlights the use of 'Skill' concept. An insight description on how the concept of 'Skill Recipe' can be used to execute the equipment 'Skills' to fulfil the product's assembly requirements is also provided. This is viewed as a critical concept to achieve high performance in 'Plug-and-Produce'. To translate the base semantic definitions, we have developed new libraries that are fully compliant with the AML standard. The main purpose of using AML in this context is to bridge production and other engineering domains. An overview of the literature that covers the past and current trends in data exchange and standards is presented, while pointing out the existing challenges and limitations. The vision of this paper is to support the standardization effort of integrating information for design, build, ramp-up and operation of production systems. Hence, this approach elucidates the use of existing AML concepts to model and instantiate Product, Process and Resource (PPR), and the underlying definitions such as: 'Skills', 'Skill Recipes' and 'Skill Requirements'. Finally, this paper illustrates the implementation of this approach in AML with a help of an industrial case study demonstrated within the openMOS project.***

Keywords: AutomationML; 'Plug-and-Produce'; Cyber Physical Systems; semantic models; Product, Process and Resource; Skill; Skill Recipe; Skill Requirement

I. INTRODUCTION

The vision of the 'open Dynamic Manufacturing Operating System for Smart 'Plug-and-Produce' automation components' (openMOS) project [1] is targeted at maximizing the economic sustainability of production systems by following three main innovation strands: (1) enabling 'Plug-and-Produce' capabilities for automation equipment, robots and machines, (2) aiding horizontal and vertical communication between all hardware and software entities for innovating new business functions, and (3) creating a manufacturing operating system (MOS) that is easily extendable and adaptable towards the introduction of new products, work orders and equipment modules, which envisages easy deployment, optimization and changeover management strategies.

In terms of the assembly domain, the existence of standards will be critical to achieve adaptable systems by guaranteeing compatibility between assembly equipment modules. The effort towards standardization is collaborative and mutually beneficial rather being limited to any individual company or an organization [2]. The standardization of these models is based on the maturity of the domain knowledge that ensures interoperability, integration and acceptance of the technology [3]. Within the domain of manufacturing automation, numerous standards have been developed by ISO, ISA and IEC. Due to the inherent and complex nature of this domain, most of these standards are part of lengthy series and difficult to implement. Nonetheless, the effort towards standardization is critical for the realization of the current drive towards CPS. The vision requires a clear, transparent and scalable 'language' and model that facilitates the integration of existing and emerging technologies across various engineering domains.

This paper proposes an AML based model that formalizes the use of 'Plug-and-Produce' components and creates the basic elements that supports CPS. The proposed approach builds on the existing AML concepts and makes it open for the integration of current legacy systems with other engineering domains, while enabling the next generation of CPS. The use of AML for the assembly domain and its wider industrial acceptance will be a positive move towards standardization.

II. LITERATURE REVIEW

Along with the rest of the society, production systems are also becoming more dependent on the implementation of the advances in information and communication technologies (ICT). 'Plug-and-Produce' was proposed to simplify the way assembly systems are built. This means equipment should be enabled with automatic discovery and registration of machine topology, along with capability orchestration by communicating with their networked partners [4]. This leads to more self-aware, agile and responsive equipment modules which are able to deal unplanned breakdowns and rapid fluctuations in production requests [5]. There is an expectation that manufacturing enterprises should be able to receive information from their stakeholders and operational units much quicker to act/react faster. However, the production landscape is complex with high interdependence between of equipment supplies, system integrator and end users. Quick responses to the volatile market demands has become more challenging with respect to the growing demand towards highly customized products, which exponentially increases the number of possible variations (different colors, shapes, features, etc.) for the same product. This subsequently increases the complexity of designing assembly lines in a short period of time.

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

To tackle some of these challenges, quite a few research projects such as: EUPASS [6], IDEAS [7], Transparency [8] and ReBorn [9] were focused on addressing some of the inherent problems within the assembly domain. The outcome from such projects are mainly related to the formalization of the product to be assembled, the definition of processes to assemble the product and the description of the actual physical system that executes the assembly processes. This includes the detailed description and information mapping between Product, Process and Resource (PPR) domains to facilitate the automated generation of process sequences and system configurations for production requests [10].

The 'Skill' concept that is based on the IEC 61499 standard [11] served as the base for building the domain models in the above mentioned projects. This alone would not be sufficient to establish the relationships, dependencies and other underlying constraints that will support the design and operation of production systems. The definition of 'Skill Requirements' facilitates the formalization of the product work flow requirements; special handling needs and other business objectives. The 'Skills' define the functional capabilities of the equipment modules, while the 'Skill Recipe' is an instance of 'Skill' with varying granularity, and it is specially catered to fulfill the various parametric requirements of the 'Skill Requirements'. The recurrent use of these concepts and definitions reflects that the use of standards for this domain is not alone a good practice, but an essential tool for creating reliable systems and repeatable processes to support the next generation CPS [12]. In this direction, quite a few engineering process libraries and best-practice guidelines are established by the research projects and standardization organizations. A few popular and the most resent standards are:

- VDI guideline 2221 "Systematic approach to the development and design of technical systems and products" [13]
- VDI guideline 5200 "Factory Planning" [14]
- VDI guideline 2206 "Development methodology for mechatronic systems" [15]
- Factory Design and Improvement (FDI) model [16]
- ISA-88 the hierarchical model for manufacturing control architecture [17]
- ISA-95 automated interface between manufacturing enterprises and control systems [18]

The various data formats applicable for modelling mechatronic systems are: the '*STandard for the Exchange of Product Model data*' (STEP) – ISO 10303 [19], '*Jupiter Tesselation*' (JT) – a data format for geometry representation defined within ISO 14306, *'XML Process Definition Language'* (XPDL) – is an XML dialect for business process modelling, and *'XML Metadata Interchange'* (XMI) - is a data format for exchanging meta data standardized by *'Object Management Group'* (OMG) popularly known for *'Product Lifecycle Management'* (PLM) products. In this landscape, data exchange formats like AML and STEP can be a major game-changer by being able to cover all or at least most of the key information within the engineering processes of the production systems [20]. [16] has conducted a detailed analysis of various standards within the PPR domain and presented it as shown in **Table 1**.

Table 1: Summary of standards relevant for enabling PPR data exchange [16]

PPR Category	Standards
Product data	DXF, DWG, CGM, HPGL, IGES, STEP AP203, STEP AP214, JT, VRML, X3D, STEP AP239, AP242 and the OMG PLM Services
Process data	OAGIS, ANSI/ISA-95, MTConnect, PSL
Resource data	CMSD, B2MML, STEP AP239 and the OMG PLM Services

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 [21]. AML uses CAEX, a top-level and neutral data format providing the interface for well-established data formats in engineering aspects that includes topology, geometry, kinematics, behavior and sequencing information. The main aim of AutomationML is to join engineering tools of different domains such as; process control engineering, mechanical plant engineering, electrical design, process engineering, robot programming, PLC programming, HMI development etc. The modelling features of various data formats was analyzed by [22]. **Table 2** is an aggregated representation of [22]'s detailed study.

Table 2: Requirement fulfilment of data formats (based on [22])

Features	Data Formats				
	AML	STEP	JT	XPDL	XML
Mechanical Data	+	+	+	-	-
Electrical Data	+	+	-	-	-
Process Control Data	+	-	-	+	-
Topological Information	+	+	+	-	+
Establishing Concept Relationships	+	+	-	-	+
Tracing Concept Dependencies	+	+	-	-	+

AML provides a common language for different engineering tools and links for detailed engineering information, which is easily exchangeable between multi-engineering disciplines and projects. Nevertheless, AML as some limitations that should be taken into account for effective and efficient description of PPR engineering domains. The main difficulties listed by [23] in terms of using of AML are as follows:

- Complex data structures with intricate links between disciplines.
- Difficulties in the integration of AML with different disciplines internally and with external data.
- Limited support for cross-disciplinary analytics by the end users.
- Limited options for platform independent browsing of AML data.
- Support tools for dealing with the above points as well as modeling and assessment of changes.

This paper focuses on defining new libraries within AML that facilitates the creation of 'Skill Requirements', 'Skills' and 'Skill Recipes' for the self-description of equipment modules to support high performance in 'Plug-and-Produce'.

III. Problem Definition

The self-description provides all the physical and functional specifications of an equipment module for the creation of a virtual entity, which will be a digital replica of that equipment module. In terms of 'Plug-and-Produce' devices, they should have embedded information about their process capabilities (Skills), data that it gathers during the execution of the processes, and other information relevant to its maintenance and diagnostics details. This is critical for CPS, since these systems would require an exact virtualization of actual physical entities to deliver their vision. In this direction, AML is viewed as the prime candidate to aggregate all the information of a physical entity (electrical, mechanical, geometry, etc.) in a coherent structure that can be easily interpreted across multiple engineering domains and tools.

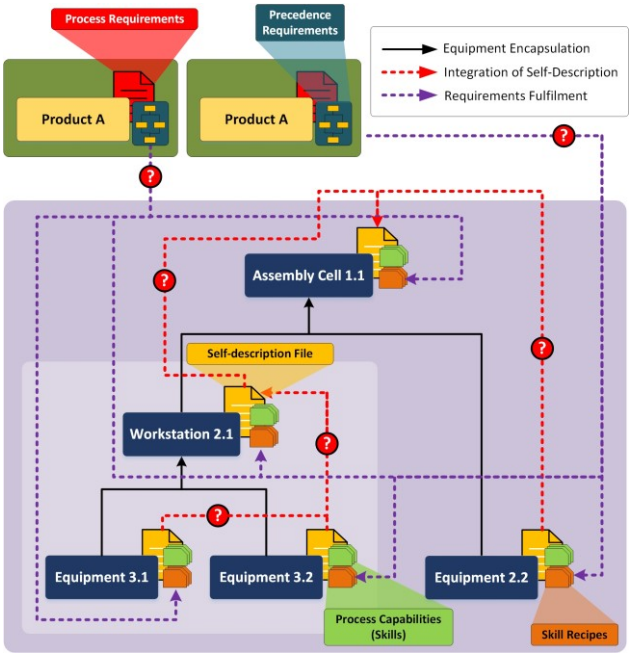

Figure 1: Overview of Problem Definition

An AML file can be an aggregation of several self-descriptions of equipment modules, which can be used to represent a hierarchical system or sub-system. The information stored in this AML file can be used to provide an overview of the same system in the cloud (cyber-side), which precisely can display a list of functional capabilities ('skills') that a workstation or an assembly cell can perform. This correlation between the cyber and the physical sides can help the user to virtually design, simulate and evaluate several management strategies. Therefore, the cyber representation of a physical system can help to emulate via monitoring, event forecasting and optimization based on the current and past data archives. This needs to support the system integration and exchange of information throughout the several phases of assembly system development such as: design, build, ramp-up, production and change (reconfiguration) phases.

One of the significant challenges in this respect is to formalise the structure of the self-description concept in way that it is compliant with existing sematic models targeting CPS. Another major challenge lies in the aggregation of self-descriptions, which can be of varying levels of system topology. It is important to note that the self-description includes the process capabilities of the equipment modules, but it does not provide the information on how to execute these capabilities to realize a product. This will require several links between the 'Skill Requirements' of the product and the actual execution information from the equipment module's 'Skill Recipes'. **Figure 1** provides an overview of the challenges with the integration and mapping of information across the system. The self-description files require relationships between the PPR domains as well as information related to equipment granularity.

IV. The AML Model

The primary objective of this model is to use as much of possible concepts from AML and enhance them by integrating 'Plug-and-Produce' concepts enabling the seamless use of AML for CPS. Toward this end, an existing semantic model will be used as a baseline for the aggregation of all the various phases of development and operation of assembly systems. The aim is to ensure that the AML model can be used by the assembly system, its components, and other engineering tools and domains that are under the roof of CPS. The model also incorporates the Product, Process and Resource (PPR), as it is a core concept of AML, that is used to establish the links, which ultimately allow the assembly system to execute processes to create a product or its sub-part. The main motivation for using AML is to ensure interoperability with other tools, which AML already delivers [21].

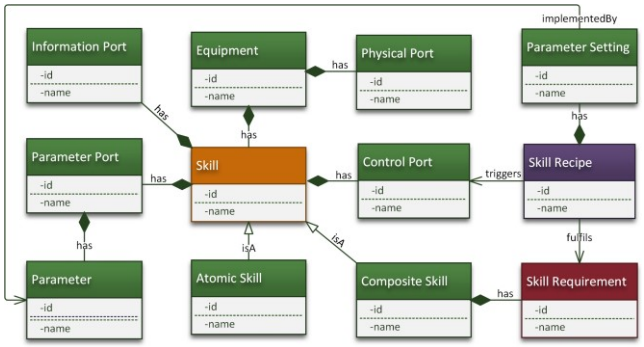

Figure 2: Overview of the Semantic Model

The semantic model established in the openMOS project [1] serves as the baseline for enhancing AML with the necessary concepts required to enable 'Plug-and-Produce' while ensuring that the AML core concepts are respected. The semantic model was developed to enable 'Plug-and-Produce' via the analysis of the relationships between the core concepts, such as: 'Skill', 'Skill Requirement' and 'Skill Recipe', with an underlying common concept of 'Skill Type'. 'Skill' defines the process capabilities of the assembly system and its sub-systems or 'Equipment Modules'. The 'Skill Requirements' enable the formalization of the product, or other equipment's, needs and business objectives. Both these concepts need to share a common 'Skill Type' if automatic matching between them is expected. The next step enables the base for system to operate, which is the definition of how the 'Skill' will be executed to fulfil a 'Skill Requirement'. This requires the creation of a 'Skill Recipe', which not only establishes the traceability between

these two concepts, but also is used to formalize the necessary parameters for the execution of 'Skills' as shown in **Figure 2**. This is particularly important in real systems, as these tend to be tweaked in the ramp-up stage, until the system is setup for production.

As mentioned above, the 'Skill Recipe' concept defines how a 'Skill Requirement' can be fulfilled by an equipment's 'Skill', which has an acceptable range of 'Parameters'. In AML, the 'Skill Recipe' is an instantiation of a 'Skill' with specific 'Parameter Settings' that can fulfill one or more 'Skill Requirements'. The 'Composite Skill' concept provides the means to deal with varying degrees of process capability granularity. The proposed model takes advantage of the already defined 'Skill Requirement' to establish higher level compositions, which can be fulfilled by lower level 'Skill Recipes'. It is important to note that there can be multiple 'Skill Recipes' for each 'Skill Requirement'. Similarly, there can be multiple 'Skill Requirements' for those 'Skill Recipes'. This means we can decouple all 'Equipment Modules' and their 'Skill Recipes', to use them as a part of aggregated 'Composite Skills', or use the same 'Skill Recipes' to match the product requirements.

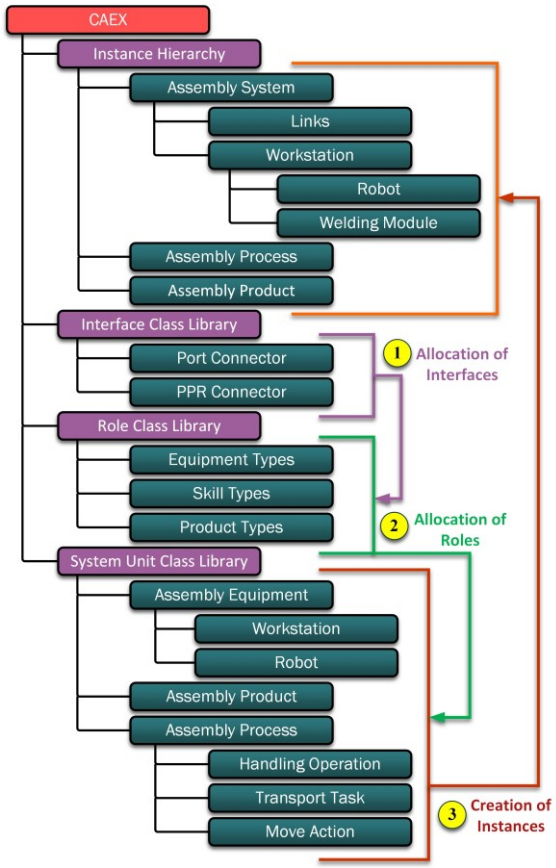

Figure 3: Overview of the Basic AML Concepts and Relationships

The execution of assembly processes are event based and it uses the 'Control Port' of the 'Skill' to trigger the start of the execution of Skill Recipes. Other important ports are the 'Information Ports' as these can be used to include some of the KPIs, such as execution time or energy consumption. These are intended for later use towards the performance measurements and optimizing the execution of assembly processes. The equipment's 'Physical Port' establishes physical interfaces with other equipment modules. An interface is defined by the means of two ports, where the ports are of the same type and compatible port directions.

The model also uses the AML core structures, namely the following libraries: System Unit Class Library, Role Class Library and Interface Class Library. The system unit class library serves as a platform of templates for creating specific classes for 'Equipment Modules', 'Physical Ports', processes classified as 'Skills' and other concepts within the assembly domain. The role class library facilitates the creation of specific types for 'Equipment Modules', 'Skills', 'Ports' and other type specifics for concepts within this domain. The interface class library helps to define the various types of interfaces such as: 'Physical Connections', 'Material Flow Connections', 'Precedence Connections' and others that are defined within this domain. These concepts can be aggregated to define the necessary concepts which can in turn be used to create 'Instance Hierarchies' for assembly system requirements, equipment modules and system configurations. **Figure 3** gives an overview of the main concepts added as well as how the libraries are interrelated.

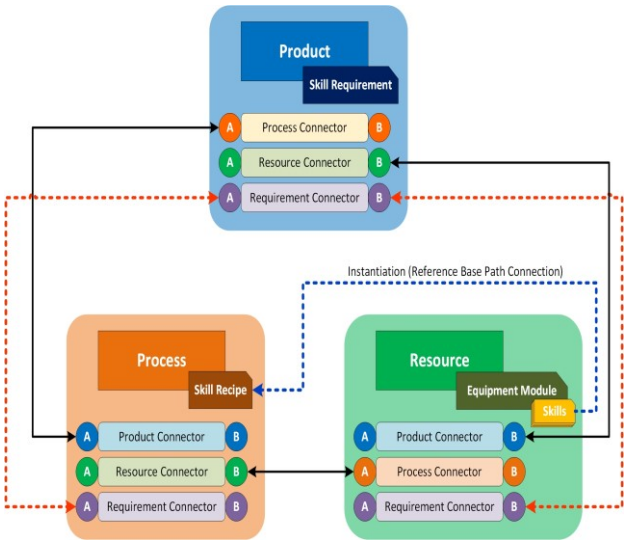

Figure 4: Overview of the AML Connections

One of the key feature of AML is that it supports the mapping of information to establish relationships and dependencies. In this sense, this model defines a few new interface classes in AML such as: 'Requirement Connector', and 'Precedence Connector', while making use of the existing 'Port Connector' and 'PPR connector'. The 'Requirement Connector' is used to map the connections between the 'Skill Recipe' that fulfils a specific 'Skill Requirement' and the 'Equipment' used to fulfil that 'Skill Requirement'. 'Precedence Connectors' provide links which express the sequential precedence relationships between the 'Skill Requirements'. The 'Port Connector' helps to map the physical interface between two 'Equipment Modules'. The 'PPR Connector' maps the links between the product to its process specifications and the processes as shown in **Figure 4**. A detailed version of this approach will be available at [1] as an open access document.

V. Illustrative Example

The illustrative example for the proposed model is based on a demonstrator developed within the openMOS project [1]. The demonstrator provides a scenario to illustrate the use of AML to instantiate a simple assembly cell consisting of self-descriptive 'Plug-and-Produce' equipment modules and to highlight how these can be linked to higher level information that enable the system to operate. The demonstrator is a robotic assembly cell developed for testing and training purposes. This is comprehensive enough to demonstrate the complexity of the system, and is compliant with current OEM standards used within the automotive industry. The cell breakdown is shown in **Figure 5**. The cell is used for spot welding of a car parts (herein referred as the 'products' shown in **Figure 6 (B)**). The setup of the assembly cell includes a workstation consisting of a 6-DoF robotic arm, a welding module and a vision system for inspecting the quality of the weld. The product is delivered to the assembly cell according to the production needs via an Automated Guided Vehicle (AGV). The main purpose of the demonstrator in terms of the project is to display the developed execution approach and integrate legacy systems with the recent 'Plug-and-Produce' concepts.

The first step in using AML to instantiate a hierarchical system is to define the system interfaces, which helps to map the relationships between different entities and concepts in the model. In the current use case, we are mainly dealing with interfaces, namely: 'Requirement Connector', 'Port Connector', 'Precedence Connector' and 'PPR Connector'. The second step is to define the classifications and the functional behavior of the assembly equipment, which ideally should already exist as it defines the system taxonomy. Here, in addition to AML base role class library, a few case specific role classes were created such as: equipment types, skill types, port types, parameter types, etc. Also, all the role classes should be assigned with appropriate interfaces as shown in **Figure 3**. It is important to note that these definitions (type specifications) should ideally be the same for the entire domain of assembly systems, but it is permissible to have project specific definitions to enable reusability in restrictive environments.

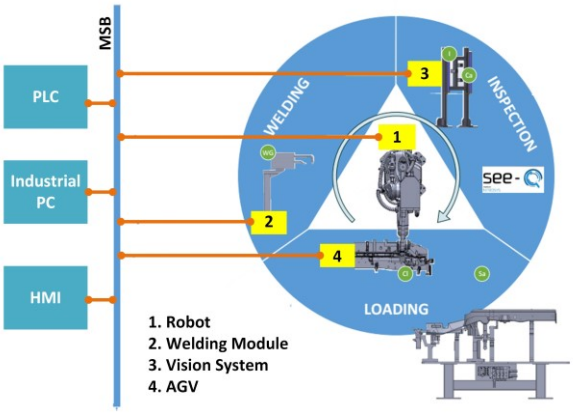

Figure 5: Overview of the Demonstrator Setup

Both the system interfaces and the role classes are expected to be less changeable elements of the system, as these provide overarching definitions for the systems. The third step in is to create the system unit classes in accordance with the rules and guidelines defined in the semantic model. This step involves the allocation of roles to each of the newly created system unit classes. It is important to note that each element in the system unit class can have one or more roles assigned to it, as it might fulfil multiple roles in the system. AML also provides the feature 'Role Requirement' to ensure that the class has a mandatory role.

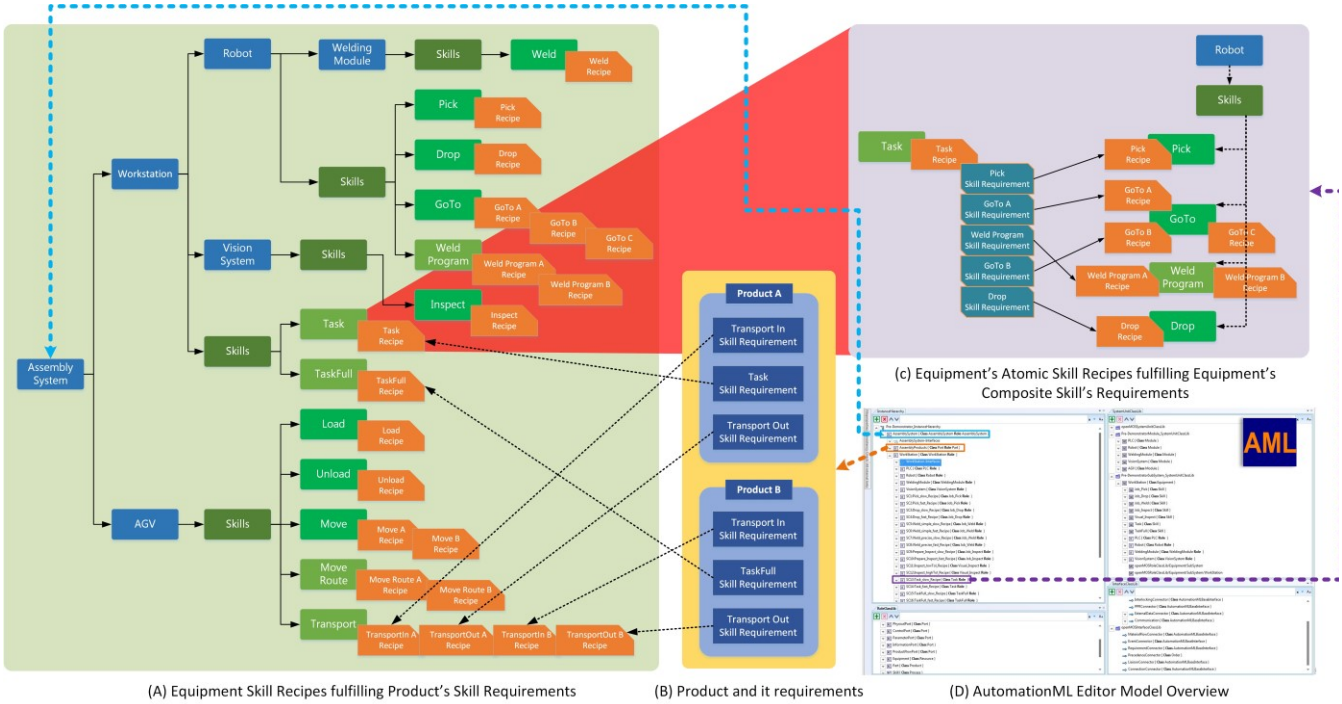

Figure 6: Overview of the Implementation of PPR Relationships in AML

Once the three libraries of AML are filled with appropriate definitions, the instantiation of the system can be initiated by dragging and dropping (using AML editor) appropriate system unit classes into the newly created instance hierarchy. Finally, it is necessary to establish the dependencies and relationships between various instances by connecting the appropriate interface ports. This will detail the physical (equipment configuration) and logical (process configuration) configuration of the system.

The demonstrator modules include specific skills as shown in **Figure 6(A)**. The concept 'Skill' is categorized into 'Atomic Skill' and 'Composite Skill'. For instance, 'Pick and Place' is a composite skill encapsulating three atomic skills, which can be defined as: 'hold', 'move' and 'release'. For simplification, any skill which cannot be broken down is classified as atomic. The robot controls the welding module and therefore encapsulates the welding module inside it. The welding module includes a simple 'weld' (Atomic) skill, while the robot has 'pick' (Atomic), 'drop' (Atomic), 'go to' (Atomic) and 'weld program' (Composite) skills. The vision system has an 'inspect' (Atomic) skill. For each 'Skill', there can be multiple 'Skill Recipes', but most cases have one 'Skill Recipe' per 'Skill', as they have no variability. The workstation on the higher-level exposes two skills, 'task' (Composite) and 'task full' (Composite). The 'Skill Recipes' for these composite skills includes 'Skill Requirements', which will be fulfilled by the equipment's atomic recipes as seen in **Figure 6(C)**.

One of the key aspect of this contribution is on mapping the information relevant to the 'Skill Recipes' and how this is aggregated in a hierarchical system. It is important to note that every instance of an 'Equipment Module', including its related AML libraries, is the representation of an individual self-description file, which can then be aggregated to create the self-description of the system as shown in **Figure 6(A)**. These files will include the 'Skill Recipes', which are instances of the 'Skills' as described in Section IV. **Figure 6 (C)** illustrates the concept of 'Composite Skill' providing an overview of how its 'Skill Requirements' are linked to the 'Atomic Recipes', following the proposed approach in Section IV. The product's requirements shown in **Figure 6 (B)** are matched with an appropriate 'Skill Recipe' that has a specific 'Parameter Setting' that can fulfill a 'Skill Requirement'. The 'Skill' executes the 'Skill Recipe' when the appropriate 'Control Port' is triggered by the arrival of the product to the workstation. **Figure 6 (D)** provides the aggregated AML file overview using the AML editor. This demonstrates the ease of decoupling elements as AML is XML based. This aggregated information can support higher-level decision making, while upgrading or changing the system.

VI. Conclusion

The paper provides insight into the potential use of AML to be extended to support 'Plug-and-Produce' concepts. This would mean a seamless integration with other engineering domains as AML already provides support for multiple tools. The basic principles behind the AML enhancements is illustrated with the help of an industrial case study used to demonstrate the activities of the openMOS project. This also exhibits the potential of AML as self-description files, which can be aggregated to generate the overall system description.

However, the instantiation effort for such models is quite tedious and time consuming, as AML does not have sufficient supporting tools. Additionally, the possibility of human error and maintaining consistency of information over several users is very difficult. In line with this, the future work will focus on developing an Application Program Interface (API) that comforts the entire process of modelling with AML Editor and supports the validation of the enhanced AML model. Furthermore, the future work will be focused on the translation and information mapping from AML to OPC UA to enrich the 'Plug-and-Produce' capabilities by enabling auto-discovery functionality, ultimately targeting the use of this information for system execution.

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

[1] "openMOS - Open Dynamic Manufacturing Operating SYstem for Smart Plug-and-Produce Automation Components."
[2] Y. Koren, S. J. Hu, P. Gu, and M. Shpitalni, "Open-architecture products," *CIRP Annals - Manufacturing Technology*, vol. 62, no. 2, pp. 719–729, 2013.
[3] P. Ferreira, "An Agent-Based Methodology for Modular Assembly Systems," *University of Nottingham*, no. April, 2011.
[4] S. Wassilew, L. Urbas, J. Ladiges, A. Fay, and T. Holm, "Transformation of the NAMUR MTP to OPC UA to allow Plug and Produce for Modular Process Automation."
[5] OECD, "Future Factory," *OECD Economic Surveys: Ireland 2011*, pp. 8–9, 2011.
[6] "European Commission : CORDIS : Projects & Results Service : Evolvable Ultra-Precision Assembly Systems." [Online]. Available: http://cordis.europa.eu/project/rcn/75342_en.html.
[7] "IDEAS project." [Online]. Available: http://www.ideasproject.eu/.
[8] "Transparency Project." [Online]. Available: http://www.transparency-project.eu/index.php?transparency.
[9] "ReBorn Project." [Online]. Available: http://www.reborn-eu-project.org/.
[10] B. R. Ferrer, B. Ahmad, A. Lobov, D. A. Vera, J. L. M. Lastra, and R. Harrison, "An approach for knowledge-driven product, process and resource mappings for assembly automation," *IEEE International Conference on Automation Science and Engineering*, vol. 2015–October, pp. 1104–1109, 2015.
[11] L. H. Yoong, P. S. Roop, Z. E. Bhatti, and M. M. Y. Kuo, "IEC 61499 in a Nutshell," *Model-Driven Design Using IEC 61499: A Synchronous Approach for Embedded and Automation Systems*, pp. 1–194, 2015.
[12] P. Ferreira and N. Lohse, "Configuration model for evolvable assembly systems," *CIRP Conference on Assembly Technologies and Systems (CATS) 2012*, pp. 75–79.
[13] VDI, "VDI 2221 Methodik zum Entwickeln und Kosntruieren technischer Systeme und Produkte," *Verein Deutscher Ingenieure*. pp. 1–44, 2013.
[14] VDI-Gesellschaft, "VDI 5200 - Fabrikplanung Planungsvorgehen," *VDI-Handbuch Produktionstechnik und Fertigungsverfahren*, vol. Band 1: Gr, pp. 1–24, 2011.
[15] V. D. I. VDI 2206, "VDI 2206 - Entwicklungsmethodik für mechatronische Systeme," *Design*. p. 118, 2004.
[16] S. Choi, K. Jung, B. Kulvatunyou, and K. C. Morris, "for Designing Smart Manufacturing Systems," vol. 121, pp. 422–433, 2016.
[17] A. N. Standard, *Batch Control Part 1 : Models and Terminology*, vol. 1, no.1995.
[18] D. Standard, "DRAFT STANDARD Enterprise-Control System Integration Part 1 : Models and Terminology," 2008.
[19] M. J. Pratt, "ISO 10303, the STEP standard for product data exchange, and its PLM capabilities," *International Journal of Product Lifecycle Management*, vol. 1, no. 1, p. 86, 2005.
[20] X. Xu and A. Y. C. Nee, *Advanced Design and Manufacturing Based on STEP*. 2009.
[21] AutomationML e. V., "Whitepaper AutomationML Part 1 – AutomationML Architecture State : May 2012," no. May, pp. 1–80, 2012.
[22] A. Luder, N. Schmidt, R. Rosendahl, and M. John, "Integrating different information types within AutomationML," *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, 2014.
[23] M. Sabou, F. Ekaputra, O. Kovalenko, and S. Biffl, "Supporting the engineering of cyber-physical production systems with the AutomationML analyzer," *2016 1st International Workshop on Cyber-Physical Production Systems, CPPS* 2016.