An open distributed architecture for flexible hybrid assembly systems: a model-driven engineering approach

Kleanthis Thramboulidis

Abstract Assembly systems constitute one of the most important fields in today’s industry. The need for flexibility in the assembly process imposes several challenges in the development of this kind of systems. Model-driven engineering (MDE) has been successfully used to alleviate the complexity of platforms and express domain concepts effectively. In this paper, an open distributed architecture for the engineering of evolvable flexible hybrid assembly systems is described. The proposed architecture is based on the model-driven development paradigm. Models are used to represent in a formalized way the structure, the behavior, and the requirements of the assembly systems. A domain-specific engineering tool is defined to facilitate the assembly system engineer in the engineering process of assembly systems. Specific meta-models are defined to capture domain knowledge and guide the engineer in the construction of the models required to construct the assembly system. The proposed approach imposes a correct-by-construction approach and is based on an architecture that exploits the benefits of Cyber Physical paradigm utilizing web technologies, Internet of Things, and Cloud computing.

Keywords Assembly system engineering · Evolvable assembly systems · Model-driven engineering · SOA-based engineering tool · Meta-modeling

1 Introduction

For a manufacturing product to be produced in its final form, its constituent parts should be properly assembled. This process is well known as Assembly Process (AP). The entity that performs the assembly process is the Assembly System (AS).

Automation technology has considerably increased the effectiveness of ASs. However, as product variety increases due to the shift from mass production to mass customization, ASs must be designed and operated to handle such high variety. A review of state-of-the-art research in the areas of AS design, planning, and operations in the presence of product variety is given in [1]. Methods for assembly representation, sequence generation, and assembly line balancing are reviewed, and the authors conclude that many opportunities exist for future research in this domain. This is also claimed in [2] where the author admits that “many works have addressed the research topic of assembly systems design in support of high product variety, but still a lot of research opportunities could be identified.” Especially for product variety in the context of highly volatile markets, the author claims that a comprehensive framework is still missing. Among the challenges highlighted in [1] in the ASs domain, the following were picked out:

1. A new approach to provide manufacturing engineers with more comprehensive information with convenient data management features. The current assembly representations are considered limited in terms of the comprehensiveness of assembly information. Bill of Material (BOM) cannot directly represent the complex physical assembly processes, and liaison graphs are considered as not suitable in representing hierarchical functional structures.
2. An assembly representation enabling interoperability across different locations and software platforms is required.
3. Determination of all possible assembly sequences as this greatly affects the total design process of a product.

An open distributed architecture for flexible and evolvable hybrid AS (HAS) is described in this paper. The term hybrid is used to refer to ASs that include both human and machine assemblers, while the term flexible is used to refer to ASs that may change their behavior based on different assembly process descriptions. It is argued that the proposed architecture addresses the above challenges and establishes a new approach for the correct-by-construction engineering of systems in this domain. Moreover, this architecture facilitates the exploitation of technologies such as Internet of Things (IoT), Cloud computing, and Cyber Physical Systems in the domain of flexible and evolvable hybrid assembly systems.

The key concept of the proposed architecture is the use of meta-models to formalize the domain knowledge and facilitate the job of the AP engineer. This is highly required since the increase of structural complexity and part quantity made the manually planning of the assembly process a hard work even for skilled engineers [3]. The proposed meta-models act as a kind of domain-specific language to elevate the complexity of the assembly system platform by effectively expressing domain-specific concepts.

The approach presented in this paper does not consider BOM and liaison graphs as assembly representations, not even it uses these terms. Alternatively, it discriminates between structural and behavioral information and captures this knowledge in three separate evolvable models, namely the product’s structural model, the assembly process model, and the assembly system platform model. The knowledge related with the construction of these models is captured in the corresponding meta-models. The use of meta-models leads to a correct-by-construction approach in assembly system development that is more effective compared to the traditional tedious construct-by-correction one [4]. Ontologies may be utilized for a machine-readable representation of the proposed framework, and the Cloud could be used to establish a distributed architecture for ASs. Web services are an effective communication mechanism to support a low coupling between the architecture elements. However, a more traditional distributed architecture based on technologies such as RPC or RMI may be utilized if the introduced high coupling between the elements of the architecture is not an issue.

The main objectives of the proposed architecture are (a) to have all the benefits of the automation technology preserving the flexibility of human-based ASs and (b) to increase the effectiveness of the development process of ASs. This facilitates the development of a new generation of automated hybrid ASs that are characterized by flexibility [5] and evolvability [6]. The term flexible is used to refer to the infrastructure that will enable the eight types of flexibility defined and described in [5] and [7]. Based on [5], machine and process flexibility as well as routing and expansion flexibility are among the different types of flexibility. Since some researchers claim that “the acquisition of flexible technology as a direct response to changing markets is not necessarily the panacea it is widely believed to be” [7], it is a challenge to propose an approach that would limit the factors against the adoption of flexibility and this was one of the motivations of this work. Authors in [8] introduce the concept of fully flexible assembly systems as a new generation of assembly systems.

Model-driven engineering [9] has been adopted to address (a) the need for flexibility and evolvability and (b) the complexity of the assembly process. Specific meta-models are defined to capture the domain knowledge and formalize the development process of ASs. The contribution of this paper is to define a set of meta-models and a reference open distributed architecture that exploits these meta-models for the engineering of evolvable hybrid assembly systems. Meta-models for the product, the assembly process, and the assembly system platform are presented, and their use in the development process of flexible and evolvable hybrid ASs is described. The proposed approach is in the context of the Model-Based Systems Engineering (MBSE) as defined in [10].

The term open architecture is used to refer to the ability to use any combination of standard-compliant components in the design and evolution process of the assembly platform. An open architecture provides the infrastructure that is required for the easy integration of heterogeneous workers and other constituent components of the assembly platform. In addition, by adopting the Representational State Transfer (REST) architectural style [11] for the integration of the components, the proposed solution inherits all the benefits of loose or even zero coupling among its components. By employing the open architecture paradigm, the assembly engineer would be able to choose the workers and the other constituent components that best meet the specific needs of the assembly platform and still remain confident that the resulting integration will satisfy requirements. Furthermore, legacy systems could be integrated in an open architecture assuming that the information required to develop the appropriate wrapper is available from the manufacturer of the legacy system.

The remainder of this paper is structured as the following. In Section 2, background and related work are presented and discussed. Section 3 describes the proposed reference architecture and its key elements. Key concepts are identified, and basic terminology is established. The proposed architecture for the HAS Engineering Tool is briefly presented. Section 4 presents the proposed meta-models in this work that formalize the knowledge required for the engineering process of evolvable hybrid assembly systems. Meta-models for product, assembly process, and assembly system platform are presented and discussed. Finally, the paper is concluded in the last section.
2 Background and related work

The work presented in this paper is based on Model Integrated Mechatronics (MIM) [12] and 3+1 SysML-view model [13], which constitutes an approach for applying model-driven engineering [9] on the mechatronics systems domain.

Other research groups have also reported on the successful use of this paradigm in mechatronic systems, e.g., [14]. MIM adopts model-driven engineering (MDE) and proposes a framework for its application in the domain of mechatronics and in particular in Manufacturing systems engineering. The Mechatronic Component (MTC) is defined as the key construct in the development of Manufacturing systems. The Manufacturing system is defined as a composition of MTCs, which are integrated to collaborate with the objective to provide the system level behavior required to satisfy stakeholder needs. Those needs include among others, functional requirements, quality of service characteristics (QoS), and an optimal use of available resources. The MTC is defined as the element of the mechatronic system that is composed of a mechanical part, an electronic part, and a software part, which are tightly integrated. An MTC appears as an entity with well-defined structure and behavior and is characterized by provided and required services offered through well-defined interfaces. The MTC construct is ideal for the representation of assemblers as well as the other elements of the assembly platform.

The 3+1 SysML-view model is a realization of MIM based on the System Modeling Language (SysML) [15], which is described as a general purpose graphical modeling language for specifying, designing, analyzing, and verifying complex systems [10]. More specifically, SysML is used as the primary artifact (a) for modeling the mechatronic system in the mechatronic layer of the MIM architecture and (b) for constructing the core model of the whole system [13]. In this paper, the 3+1 SysML-view model is exploited for the modeling of assembly systems.

The assembly system is defined in the Mechatronic layer as a composition of MTCs that represent the elements of the assembly system platform. Based on this framework, a construct is defined for the engineering of flexible and/or evolvable assembly systems. A reference architecture is described exploiting SysML and the 3+1 SysML-view model to provide the infrastructure required for the development of this kind of systems. SysML has already been used by several research groups for the modeling of mechatronic systems, e.g., [16].

Service-oriented computing [17] provides the technologies required to address the need for distributed nature of assembly systems. Ontologies can be exploited to formalize the elements of the proposed infrastructure such as product, assembly process, and assembly system platform and make them machine interpretable so that they can be more easily analyzed by the HAS Engineering Tool to assist the assembly system engineer in decision making involved in the engineering process of hybrid assembly systems. The use of Ontologies and SOAs in assembly systems has already been reported by research groups, e.g., [18, 19]. This paper does not explicitly refer to the exploitation of service-oriented computing and ontologies in the presented framework. A framework for exploiting these technologies in the engineering process of industrial automation systems has already been described in [20]. Encouraging results have been also presented by other research groups, e.g., [21].

In [22], authors describe an approach for evolvable assembly systems based on Ontologies. An assembly process ontology based on the Process Specification Language (PSL) [23] is presented. The PSL was defined according to [24] to be “a common language to all manufacturing applications, generic enough to be decoupled from any given application, and robust enough to be able to represent the necessary process information for any given application.” To this direction, it can be used for complete and correct exchange of process information among established applications. PSL does not provide assembly-specific constructs and constraints; thus, the ontology presented in [22] is considered as an extension of PSL, which is considered as a solid basis for the definition of such an ontology. However, this ontology is based on the specific structure of the assembly process based on the constructs of task, operation, and action. The task is used to define the sequence in which the components are being assembled to form the final product. The term operation is used to define the individual motions and other more hardware- and control-related activities. These three constructs correspond to three different levels of the assembly process with the task assigned to the highest one and the action to the lowest one. This approach does not address the requirements of our framework; it leads to platform-dependent assembly process descriptions since the action refers to hardware- and control-related activities, which are platform specific. This makes our approach quite different from the whole proposed structure in [22] also for reasons explained in the remainder of this paper. A robust meta-model is considered as a prerequisite for a successful ontology. Thus, the focus is firstly on the construction of meta-models that support the construction of platform-independent assembly process descriptions.

Authors in [25] propose an assembly sequence generation algorithm based on CAD data. The paper also briefly presents the most important research works towards this direction. These works can be utilized in the context of the proposed framework to get the benefits of the meta-model-driven approach. The intention of the work presented in this paper is not to propose a new algorithm for assembly sequence generation but to automatically construct a model of the assembly process that would capture the dependencies between product components as they come from the product model. This model is
independent of the assembly platform and will be the source from which the assembly platform-specific assembly process will be generated taking into account the assembly platform model. This results to the definition of the assembly sequence based on the optimal utilization of the specific platform resources which are captured in the assembly platform model. This approach is different from the one presented in [25] in that it introduces a two-step approach for the assembly sequence generation based on product and assembly platform models.

In [23], authors describe an approach called Assembly Process Micro-Planning which is based on PSL. The objective is to have the Computer-Aided Assembly Process Planning tools and Computer Aided Assembly Process Simulation Tools to use the same assembly process description. They adopt four levels of Assembly process decomposition: Assembly Job (AJ), Assembly Task (AT), Assembly Operation (AO), and Assembly Action (AA). They use the construct of AJ to model behavior assigned to a worker or a team. The construct of AT is used to model assembly sequences in AJs that correspond to the behavior of assembling a part or a subassembly. The ATs of an AJ can be executed sequentially or in parallel. This approach is considered as assembly platform dependent since the assembly platform workers are considered from the first step of AJ definition.

Authors in [26] describe a systematic approach based on process knowledge customization and meta-models for manufacturing resources. Authors in [27] describe an integrated design model of assembly systems where decisions regarding layout, assembly sequencing, task assignment, and assemblers’ locations are analyzed to obtain the optimal solution. To the best of our knowledge, there is no open reference distributed architecture based on meta-models for the engineering of evolvable and flexible hybrid assembly systems.

3 A reference architecture for evolvable hybrid assembly systems

A human-based assembly system is by default flexible, evolvable, and reconfigurable since humans may change their behavior based on different assembly process descriptions. However, traditional automated assembly systems have lost this flexibility since they are constructed with the assembly process logic hardcoded; they are constructed to perform a specific assembly process for a specific product. This is shown in Fig. 1, where the AS accepts as inputs the product’s parts and their connectors (if any) and delivers as output the finished product. Any change in the assembly logic requires a reconstruction of the assembly system that is a costly and time-consuming process. Market demands for product variations and/or product enhancements impose the need for flexible and evolvable ASs [28–30].

Flexible ASs provide generalized flexibility designed for the anticipated variations and built-in a priori. Flexibility is discriminated from reconfigurability. As defined in [31], a manufacturing system is reconfigurable when it allows “cost-effective and rapid system changes, as needed and when needed, by incorporating principles of modularity, integrability, flexibility, scalability, convertibility, and diagnosability.”

The term flexible means that the system is able to change or to do different things. In this case, the AS can be modeled by the assembly process description that captures the logic of the assembly process and an underlying platform on which this logic is deployed and executed. The term Assembly System Platform (ASP) is used to refer to the underlying platform that is a mechatronic system. The ASP accepts as input also the assembly logic in terms of an assembly process description (as part of the assembly job), as shown in Fig. 2. The assembly process description, when deployed onto the ASP, transforms it into a specific AS. The description of the AP should be in a readable and understandable by the system format for the AS to be able to realize it. The term assembly job is used to represent a job that is assigned to the assembly system; it contains information about the product to be produced, the number of items, the specific variation, quality parameters, etc.

ASs are mainly composed of assemblers and tools, both considered as resources of the assembly system platform. A flexible assembly system platform provides the functionality of not only assigning assembly behavior to its assemblers but also specifying their collaboration.

According to [6], evolvability is defined as “the ability of complex systems to co-evolve with the changing requirements, to undergo modifications of different significance, from small adaptations on-the-fly to more important transformations.” The term evolvability with the above definition overlaps with the term flexibility. A system is characterized as evolvable when it is able to change its behavior to meet changing requirements, as shown in Fig. 3.

The framework described in this paper provides the infrastructure to support both flexibility and evolvability.
3.1 Using meta-models for knowledge representation

As claimed in [1], the design of an AS requires methods to represent the assembly components and hierarchy and to generate the sequences of assembly. To this direction, authors review the most commonly used assembly representation methods, “including liaison and precedence graphs, and discuss how these methods are adopted for representation of products with variety.” BOM and Liaison graphs are reported as commonly used assembly representation methods. In the proposed approach, the structural information is discriminated from the behavioral one and two different models are used to capture this information, i.e., the product’s structural model and the behavioral model of the assembly system, called assembly process model. The information regarding the product’s structure, i.e., the product’s parts and their physical contacts of joining, called liaisons, is captured in a formalized way by the product’s structural model (PSM). Key concepts used for the construction of product’s structural models are captured in the product’s structural meta-model (PSMM) to facilitate the assembly system engineer in the construction of robust and manageable structural models of products. BOM or liaison graphs could be used as input for the generation of the PSM which will be an instance of the PSMM. The proper mapping of the corresponding meta-models, i.e., BOM and PSMM meta-models, will enable an automatic model to model transformation for the automatic or semi-automatic construction of PSMs from the BOMs.

The behavior required by a system to perform the assembly of a product is captured and represented by the assembly process model (APM). The challenge is to have an assembly system that would accept assembly requests and define dynamically its behavior utilizing the knowledge captured in the specific product’s structural model, as shown in Fig. 3.

The knowledge related with the construction of AP models is formalized in the assembly process meta-model (APMMM). APMs are defined as instances of the APMM. Two APMs defined by a two-step procedure are introduced. The first model is the platform-independent APM which captures the assembly logic required for the assembly of the final product in a platform-independent way. It captures chunks of behavior that correspond to the assembly of product’s parts without defining sequence or parallelization regarding their execution. The platform-independent APM captures just the precedence constraints among the product’s components as they are expressed in the corresponding product’s structural model and those imposed by assembly constraints imposed by other factors, e.g., product’s geometry. Thus platform-independent APM captures all possible assembly scenarios of the specific product. The second model, the platform-specific APM, describes the behavior that should be deployed to the ASP to transform it to an assembly system for the specific product. The platform-specific APM is generated by refining the platform-independent APM taking into account the structural and behavioral information that is captured in the model of the ASP. The knowledge related with the construction of ASP models (ASPMs) is formalized and captured in the assembly system platform meta-model (APMM). ASPMs are defined as instances of the ASPMM.

The use of transformation engines and generators allow the engineers to analyze various aspects of their models and synthesize, based on these models, the various artifacts for the construction of the AS platform and the corresponding simulation models. This will ensure the consistency between the implementation and the models used for the analysis and evaluation of the proposed by the AS engineer designs.

3.2 The hybrid assembly system engineering tool

For the properties of flexibility and evolvability to be realized and taking into account that an assembly system is a composition of possible heterogeneous assemblers, a uniform machine-readable assembly process description is required. For human assemblers to participate in complex assembly systems, a uniform human and machine-readable assembly process specification is required. If this is not feasible, automatic transformation between these representations should be provided. Moreover, these specifications should be easily deployed into the assembly system. An Engineering Tool, called HAS Engineering Tool, is proposed to address the need for constructing AP specifications and deploy these on hybrid ASs to perform the assembly process, as shown in Fig. 4.

The Assembly Process Engineer using the HAS Engineering Tool and utilizing the ASPMM available in the Cloud constructs the ASP model that represents the hybrid ASP that would be assigned the assembly job. The HAS Engineering Tool also allows the assembly process Engineer to construct the PSM utilizing the corresponding meta-model.

These models constitute along with the assembly process model the infrastructure that is required for generating and evaluating alternative assembly scenarios for the specific product. The specification of the selected as optimal assembly process is deployed on the hybrid ASP to construct the
specialized AS required for the assembly of the specific product or product variation.

Thus, the objective of the HAS Engineering Tool is to support the engineer to construct the optimal AS that fulfils the requirements of the stakeholders for assembling the specific product or products or perform specific assembly job(s). More specifically, the HAS Engineering Tool will allow the user

a) to capture the logic of the assembly process
b) to transform the assembly process logic into an assembly process specification for the specific platform
c) analyze and evaluate alternative scenarios
d) deploy the so constructed assembly process specification on the assembly system platform

Moreover, it will provide the functionality for modeling the assembly system platform. This model will be utilized for the implementation of the functionality that is described above.

The HAS Engineering Tool may be developed with the traditional approach of developing engineering tools. In this case, it is represented as a system, as shown in Fig. 4. However, a service-oriented architecture is considered as a more effective approach. In this section, an approach based on [20], where the infrastructure required to build a service-based Engineering Tool, is presented.

In this case, there is no one entity that constitutes the Engineering Tool but a set of distributed services that may interoperate to accomplish the engineering process. Services that would provide generic functionality as well as specific functionality required by the AP Engineer should be identified and properly defined. The definition of services in the assembly system domain is a challenge since it should be based on many parameters such as performance, flexibility, maintainability, and reuse.

Among these services, we discriminate a service to define and manage assembly jobs, a service to define and manage product’s structural models, a service to define and manage AP specifications, a service to deploy AP specifications, a service to define and manage ASP models, etc. For these services to interoperate in order to constitute a coherent HAS Engineering Tool, the constructed models are stored in the local Cloud and are identified using properly constructed uniform resource identifiers (URIs). Access to meta-models on the Cloud is through URIs.

The assembly job contains the URI of the platform-independent assembly process specification that captures the logic of performing an assembly of the final product.

3.3 The architecture of the assembly system

Figure 5 captures the core architecture of the Assembly System. Machine assemblers are interconnected through IoT while human assemblers are interconnected to the system platform through their agents in the local Cloud. Passive resources have their software representations that transform these to smart resources. Assemblers and other resources of the underlying infrastructure along with the software layer that represents the collaboration infrastructure knowledge constitute the ASP. Thus, the ASP is a distributed entity and communicates with its actors through the ASP agent. The ASP agent transforms the physical infrastructure of the AS into a smart ASP. At the lower level, the IoT already provides technologies for implementing a low and so flexible coupling between the ASP assemblers.
An interesting element of the proposed architecture is the Cyber part of the ASP that is shown in the Cloud captured in Fig. 5. The ASP agent communicates with the Cloud Cyber part of the ASP to report data regarding the operation of the ASP. The Cloud Cyber part of the ASP collects data from all the instances of the specific ASP type all over the world, and using Big Data technologies analyzes these data to produce knowledge regarding preventing maintenance, evolution, fault handling, etc. This knowledge is exploited by local ASP agents to transform every ASP instance into a self-evolvable system. This Cloud Cyber part of the ASP if properly defined and maintained will greatly improve the system maintenance, operation, refinement, evolution, etc., all those benefits that emanate from the use of Cyber Physical concepts.

4 The proposed meta-models

The key concept of MDE is to consider models as the primary artifact in the Engineering process. For the models to be effectively constructed by the assembly process engineers, a framework is required to capture the rules, constraints, and capabilities of utilizing the available resources of the assembly system platform and guide to the optimal solution. This infrastructure is described in terms of meta-models. Specific meta-models are identified and developed to capture the domain logic regarding product assembly, assembly process, and assembly system platform. The corresponding meta-models are utilized by specific model editors to construct the models that will be used in the engineering of the assembly process.

Figure 6 captures key concepts involved in the engineering process of assembly systems. An order that is given to an assembly system platform to produce a specific number of a product is considered as assembly job. The assembly job contains the identifier of an assembly process of the specific product that will be realized by the assembly system platform to perform the assembly job. The AssemblySystemPlatform is considered as a composition of one to many, i.e., [1..*], Assemblers, which is represented by the composition symbol of UML that is a filled diamond shape on the containing class end. The assembly process specification is defined in terms of actions which are the smallest reusable units of behavior that can be used to describe assembly tasks. Actions are platform independent. For an assembly process specification to be deployed on an assembly system platform, the platform should have all the skills that realize the actions used by the APS.

A product may have several assembly processes each one describing a specific process of assembling the product’s constituent parts. An assembly process is associated with a specific product or product variation, and it may be realized by one or more assembly systems, as shown in Fig. 6. Key concept of the assembly process is the assembly process description which describes the behavior that should be exposed by the assembly system to accomplish the assembly of a product. An assembly process specification is constructed using constructs of higher layer of abstraction compared to action, but
this is not shown in this concept map; it is described in the assembly process meta-model subsection.

4.1 The assembly system platform meta-model

For simple assembly tasks, the assembly system may be composed of an assembler. However, as the complexity of the assembly process increases, the assembly system is constructed as a composition of assembly subsystems each one performing a part of the assembly process, i.e., a subassembly process, and their connectors, i.e., subassembly system connectors. Figure 7 presents the proposed assembly system platform meta-model. An assembly platform subsystem that may not be partitioned into lower layer assembly subsystems is called Assembler. The Assembler is the human or machine entity with the ability to perform assembly activities. Connectors interconnect assembly subsystems through their input and output ports. Connectors may be active, such as conveyors or passive. A passive connector is just a storage location that is considered as output port for the input assembly subsystem of the connector and as input port for the output assembly subsystem of the connector. The structure of the assembly system platform is captured in a model that is instance of the assembly system platform meta-model.

4.2 The product’s structure meta-model

The product model defines the constituent parts of a product and their interconnections using the proper connectors. The product model is a prerequisite for the definition of its assembly process. For the construction of the product model, the product meta-model shown in Fig. 8 is used. Based on this, a product is considered as a composition of subassemblies (see SubAssembly block) and connectors (see Connector block). A subassembly is either a primitive part, i.e., an element of the product that is not further decomposed into other constituent parts, or a Composite part, i.e., a system element that is a composition of other elements. A connector interconnects 2 or more SubAssemblies and is associated with their corresponding Liaison. A Liaison (ConnectionPort) is used to represent a connection point or a connection surface of the SubAssembly. The optional association between Product and SubAssembly and the one between Product and Connector are used to capture product variations.
The assembly level of the composition tree in which every part belongs is captured in the SubAssembly or the CompositePart. It will be utilized to build the assembly process model.
4.3 The assembly process meta-model

In the assembly process meta-model, the assembly process is modeled as CompositionLevelAssemblyProcess. It represents in the model space the part of the assembly process that captures the assembly logic of the first level of decomposition of the product, i.e., composition level 0. This is shown in Fig. 9 as dcl-0. The aggregation hierarchy of a product is defined as consisting of $n$ levels of decomposition, which constitute the aggregation tree. The first level of decomposition, i.e., dcl-0, is the highest level of composition of the products aggregation tree. The dcl-$n$ is the lowest level of the products aggregation tree. As shown in Fig. 10, a CompositionLevelAssemblyProcess is either a PrimitiveChildAssemblyProcess or a CompositeChildAssemblyProcess. The PrimitiveChildAssemblyProcess represents in the modeling space the assembly process/subprocess that captures the assembly behavior that corresponds to the assembly of the composite part of a decomposition level in the case that all the constituent parts (child) of the composite part (parent) are primitive as defined in the product meta-model.

In the case that at least one of the child is composite, the parent’s assembly process is modeled as CompositeChildAssemblyProcess. Each CompositionLevelAssemblyProcess is characterized by the level of decomposition of the products aggregation tree. Precedence constraints are captured at all levels of the hierarchical structure of the AP, as shown in Fig. 10. They are also captured at the Action level. It should be noted that the alternative to define the association between the Operation and Action as {ordered} was not adopted since the precedence constraint at this level allows also parallelism on action execution that leads to a better utilization of the ASP resources.

A CompositeChildAssemblyProcess is defined as a composition of the assembly processes of the composite child of the corresponding decomposition level and the primitive assembly activities which are required for the assembly of the constituent parts of the parent at this decomposition level. A PrimitiveChildAssemblyProcess is defined as a composition of primitive assembly activities (PrimitiveAssemblyActivity) which are required for the assembly of the constituent parts of the parent at this decomposition level.

A PrimitiveAssemblyActivity is used to represent in the model space the process of assembling two constituent parts of the product. In case of complex parts with many liaisons, more levels of PrimitiveAssemblyActivities may be defined to represent different levels of abstraction to handle the complexity in the assembly of the parts. It is advised to define one PrimitiveAssemblyActivity for any interconnection of two parts based on the corresponding liaisons.

This low level PrimitiveAssemblyActivity is described by the set of operations that should be executed in order to complete an interconnection based on the corresponding liaisons. Thus, it is modeled as a composition of Operations. Among Operations, we discriminate assembly operations, move operations, handle operations, etc. An Operation is defined as a composition of actions. The Action is used to represent in the model space the primitive behavior of the assembly system. It has a well-defined interface and semantic. Actions are predefined and standardized to facilitate a platform-independent assembly process description. It is evident that for a platform-independent assembly process description to be realizable on a given assembly platform, the actions used for the assembly process description should be implemented in terms of skills by the assembly system platform workers.

For the product of Fig. 9, the AP is composed of one CompositeChildAssemblyProcesses and at least three PrimitiveAssemblyActivities, one for each one of the interconnections shown in the diagram as C1, C2, and C3 for the level dcl-0.

The proposed assembly process meta-model is based on four levels of process decomposition, i.e., Process, activity, operation, and action. Its structure is based on the product aggregation tree that is considered as an instance of the product meta-model. This allows the assembly process to automatically embed in its structure the composition constraints imposed by the product model. Moreover, it allows a platform-independent description that is the prerequisite for the effective application of the MDE paradigm in the assembly systems domain.

5 Conclusion

An open distributed architecture for the model-driven engineering of assembly systems has been described. The proposed architecture defines a framework for the exploitation of service-oriented computing, web technologies,
Internet of Things, concepts of Cyber Physical Systems, and the Cloud. It provides the infrastructure to address several challenges in the domain of flexible and evolvable hybrid Assembly Systems.

The assembly systems concept map addresses the communication gap that exists between production engineers and software and system engineers and establishes a basic terminology with well-defined semantics. This is the first prerequisite to a productive and effective approach to address the challenges in the multi-discipline domain of developing evolvable hybrid assembly systems.

Meta-models were used to capture the domain knowledge and are used to facilitate the engineer in the specification of the assembly process in its final executable form that may be deployed on the assembly platform. These meta-models can be used as basis for the standardization in the assembly systems domain. The assembly system meta-model can be used not only for the generation of the assembly sequence of a product but also for addressing interoperability issues in the integration of assemblers from different vendors. The introduction of two assembly process models, i.e., the platform independent and the platform specific results to a more flexible assembly process description addressing interoperability, portability, and reusability issues.

The current work focuses on the implementation of the infrastructure that is required for a prototype implementation of the proposed framework. The REST architectural style was adopted, and emerging IoT standards are utilized [32] to satisfy the requirements of the proposed framework.
References

1. Hu SJ, Ko J, Weyand L, ElMaraghy HA, Lien TK, Koren Y, Bley H, Chryssolouris G, Naas N, S iptalni M (2011) Assembly system design and operations for product variety. CIRP Ann Manuf Technol 60:715–733
2. Matt DT (2013) Design of a scalable assembly system for product variety: a case study. Assem Auton 33(2):117–126
3. Yi Liu, ShiQi Li, JunFeng Wang, Hongmei Zeng, JgPing Lu, (2014) A computer vision-based assistant system for the assembly of narrow cabin products. Int J Adv Manuf Technol, pp. 1–13, DOI 10.1007/s00170-014-6274-9
4. Balasubramanian G, Kolkhe A, Karsai G, Szipanovits J, Neema S (2006) Developing applications using model-driven design environments. IEEE Computer 39(2):33–40
5. Browne J, Dubois D, Rathmill K, Sethi SP, Stecke KE (1984) Classification of flexible manufacturing systems. The FMS Magazine 2(2):114–117
6. Hsu Y-Y, Tai P-H, Wang M-W, Chen W-C (2011) A knowledge-based engineering system for assembly sequence planning. Int J Adv Manuf Technol 55(5–8):763–782. doi:10.1007/s00170-010-3093-5
7. Bench R, Muhlemann AP, Price DHR, Paterson A, Sharp JA (2000) A review of manufacturing flexibility. Eur J Oper Res 122:41–57
8. Giulio R, Maurizio F, Andrea C, Aldo R (2013) Fully flexible assembly systems (F-FAS): a new concept in flexible automation. Assem Autom 33(1):8–21
9. Schmidt DC (2006) Model-driven engineering. IEEE Computer 39(2)
10. INCOSE Systems Engineering (2007) Vision 2020, INCOSE-TP-2004-004-02 September
11. Fielding RT (2000) Architectural styles and the design of network-based software architectures. University of California, Irvine, Dissertation
12. Thramboulidis K (2005) Model integrated mechatronics—towards a new paradigm in the development of manufacturing systems. IEEE Trans Ind Inf 1(1)
13. Thramboulidis K (2010) The 3+1 SysML view-model in model integrated mechatronics. J Softw Eng Appl 3(2):109–118
14. Burmester S, Giese H, Tichy M (2005) Model-driven development of reconfigurable mechatronic systems with mechatronic ‘UML’ in model driven architecture, vol 3599. Springer Berlin, Heidelberg
15. OMG (2012) OMG Systems Modeling Language (OMG SysML™),” Version 1.3
16. Cao Y, Yusheng L, Paredis C (2011) System-level model integration of design and simulation for mechatronic systems based on SysML. Mechatronics 21:1063–1075
17. Bichler M, Lin KJ (2006) Service-oriented computing. Computer 39(3):99–101
18. Niels L, Hirani H, Ratchev S (2005) Equipment ontology for modular reconfigurable assembly systems. Int J Flex Manuf Syst 17(4):301–314
19. Goncalo C, Colombo A, Barata J, Janmes F (2011) Service-oriented infrastructure to support the deployment of evolvable production systems. IEEE Trans Ind Inf 7(4)
20. Thramboulidis K, Doukas G, Koumoutsos G (2008) A SOA-based embedded systems development environment for industrial automation. EURASIP J Embed Syst. Article ID 312671, 15 pages, doi:10.1155/2008/312671
21. Onori M, Lohse N, Barata J, Hanisch C (2012) The IDEAS project: plug & produce at shop-floor level. Assem Auton 32(2):124–134
22. Lohse N, Hirani H, Ratchev S, Turitto M (2005) An ontology for the definition and validation of assembly processes for evolvable assembly systems. ISATP, Proceedings of the IEEE International Symposium, pp. 242–247
23. Liang D, Yuan L, Jian-feng Y, Jie Z (2009) A cooperative method between assembly process planning and simulation for complex product. International Conference on Interoperability for Enterprise Software and Applications, China
24. Craig S, Knutilla A, Ray S (1996) Unified process specification language: requirements for modeling process. NIST, National Institute of Standards and Technology, Manufacturing Engineering Laboratory, Manufacturing Systems Integration Division, Gaithersburg
25. Makris S, Pintzos G, Rentzlos L, Chryssolouris G (2013) Assembly support using AR technology based on automatic sequence generation. CIRP Ann Manuf Technol 62:9–12
26. Xu HM, Yuan MH, Li DB (2009) A novel process planning schema based on process knowledge customization. Int J Adv Manuf Technol 44(1):161–172
27. AlGeddawy T, ElMaraghy HA (2010) Design of single assembly line for the delayed differentiation of product variants. Flex Serv Manuf J 22(3–4):163–182
28. Li G-d, Zhou L-s, An L-l, Ji J-f, Tan C-b, Wang Z-g (2010) A system for supporting rapid assembly modeling of mechanical products via components with typical assembly features. Int J Adv Manuf Technol 46(5–8):785–800. doi:10.1007/s00170-009-2115-7
29. Onori M, Barata J, Frey R (2006) Evolvable assembly systems basic principles. In: Shen W (ed) IT for Balanced Manufacturing Systems 220, IFIP. Springer, Boston, pp 317–328
30. Hakan A, Ferreira J, Onori M (2013) Demand responsive planning: workload control implementation. Assem Auton 33(3):247–259
31. ElMaraghy HA (2005) Flexible and reconfigurable manufacturing systems paradigms. Int J Flex Manuf Syst 17(4):261–276
32. Thramboulidis K. “A Framework for the Implementation of Industrial Automation Systems Based on PLCs”, (under review)