A Model Driven Approach for Automated Generation of Service-Oriented Holonic Manufacturing Systems

Mohammed El Amin Tebib, Pascal Andre, Olivier Cardin

A Model Driven Approach for automated generation of Service-oriented Holonic Manufacturing Systems

Mohammed El Amin Tebib, Pascal André, and Olivier Cardin

Abstract In the context of manufacturing in Industry 4.0, software systems become of prime importance. Efficient, adaptable and trusted software services are required. Several approaches succeeded in creating a Service-oriented Holonic Manufacturing System that combines the advantages of Service-oriented Architectures and Holonic Manufacturing Systems. However these systems until now suffer from many shortcomings, among which genericity, lack of proof of the functional behaviour correctness, architecture modularity, etc. These systems are often manually implemented and become hardly adaptable and reconfigurable to different contexts (resources, workshop...). In this paper, we investigate a Model Driven Engineering approach to represent these systems in order to automate the generation of the services logic code from an abstract models and construct a new software chain that deals with all the shortcomings cited above.

Key words: Service-Oriented Architecture, Holonic Manufacturing Systems, Model-Driven Engineering, Verification

1 Introduction

Industry 4.0 increases the influence of software in the field of cyber-physical systems (CPS) where the service paradigm becomes pregnant. Cyber-Physical Production Systems (CPPS) relies on the latest and foreseeable further developments of computer science (CS), information and communication technologies (ICT), and

André. Pascal, Mohammed El Amin. Tebib
LUNAM Université, Université de Nantes, LS2N UMR CNRS 6004 2, rue de la Houssinière F-44322 Nantes Cedex, France, e-mail: pascal.andre@univ-nantes.fr

Cardin. Olivier
LUNAM Université, IUT de Nantes – Université de Nantes, LS2N UMR CNRS 6004 2 avenue du Prof. Jean Rouxel – 44475 Carquefou Cedex, France, e-mail: olivier.cardin@univ-nantes.fr

manufacturing science and technology (MST). CPPS consist of autonomous and cooperative elements and subsystems that are connected based on the context within and across all levels of production, from processes through machines up to production and logistics networks [Monostori et al., 2016]. Business processes can be connected to manufacturing processes. Cyber manufacturing is a transformative concept that involves the translation of data from interconnected systems into predictive and prescriptive operations to achieve resilient performance [Lee et al., 2016].

The unifying paradigm between processes and systems is the concept of **service**, including the cloud stack XaaS and the Service Oriented Architectures (SOA). The service paradigms supports scalability from (high level) business processes in enterprise architecture frameworks to (low level) hardware operations. The coordination and control of such complex systems by the way of actors or components requires methods and techniques to design, verify and deploy the services, possibly on the fly, making service engineering an unavoidable approach to develop new generation cyber-manufacturing systems. This is part of requirements of CPS mentioned by Wang et al. in the category 'design methodology' [Wang et al., 2015]. As mentioned by Bauer et al., the traditional automation pyramid is dissolving and manufacturing IT is moving towards service-orientation and app-orientation [Bauer et al., 2017]. As an example, in cloud manufacturing, SoA was identified to meet the requirements of all higher level manufacturing CPS layers due to the reduced time constraints present [Morgan and O'Donnell, 2015].

However "servicizing" is not sufficient to get software product with good-enough quality as defined by Meyer [Meyer, 1988]. First, we need software development techniques to improve both the quality of the resulting applications (correctness and robustness) but also its maintenance (extendibility, reusability). Second, we need analysis tools to check the service model properties on various aspects (structure, dynamics, functional and non-functional), and model transformations to compute new models or to generate code in the spirit of *Model Driven Engineering (MDE)*.

In a previous article [André and Cardin, 2018], we advocated for a new vision of CPPSs construction with an emphasis on trusted service based component systems to represent the software part of CPPSs. The main idea was to incorporate a broad set of software design techniques and practices to improve the quality of the delivered CPPSs. In this article, we follow this position when revisiting a legacy SOA oriented CPPSs application and showing by practice our vision. We mainly address the software construction process and maintenance issues (modularity, extendibility, reconfigurability, evolution) than the correctness and robustness issues which details will be postponed to future contributions. Legacy applications miss abstraction, genericity and modularity which make harder the evolution or reuse of such applications. Thanks to model driven engineering and software development best practice, we propose a software construction process that improves abstraction and separates the concerns to face the above drawbacks. Abstraction enables to reason at the good level *e.g.* verification of model properties before implementation. The separation of concerns enables to separate the problem domain from the implementation issues (simulation, real workshop, user interface, interoperability and communications...)

We present the problem statement in section 2 and sketch an overview of our proposal in section 3. Thanks to MDE, building the manufacturing software becomes itself a Software Product Line where we intend to rationalize the software production and maintenance, including features and variability, shared aspects are separated from specific aspects. The approach is illustrated by on-going experimentation on a benchmark workshop in section 4. The application of this work is discussed in section 5. In conclusion, we draw the open perspectives and expected advantages.

2 Problem Statement

The starting point is a double finding, from literature review and current practice.

Related works mention that service engineering in the context of CPPSs is still a craft activity, usually at the implementation level [Rodrigues et al., 2015, Morariu et al., 2013]. Two main levers are still required to go further. First, we need service models that can fit to various semantics and various granularity levels. Indeed, the concept of CPPS covers many classes of (physical) systems, from the manufacturing workshop to the whole supply chain. Taking the example of HMS, the control of such systems is often recursive, if not fractal, in order to aggregate the available resources and enable a heterarchic control architecture. Therefore, services that might be used at various levels of the architecture need to fit various granularity and the portability of services between different applications with their own semantic requires an adaptability of the services to be effective.

In practice, we started from the work of [Quintanilla et al., 2016] who proposed a new vision for the conception of agile HMS by coupling HMS architectures with Service-oriented ones, which gave birth to the concept of Service-oriented Holonic Manufacturing Systems (SoHMS) [Quintanilla, 2015]. They implemented a manufacturing software in Java to control an assembly line as illustrated by Figure 1. The

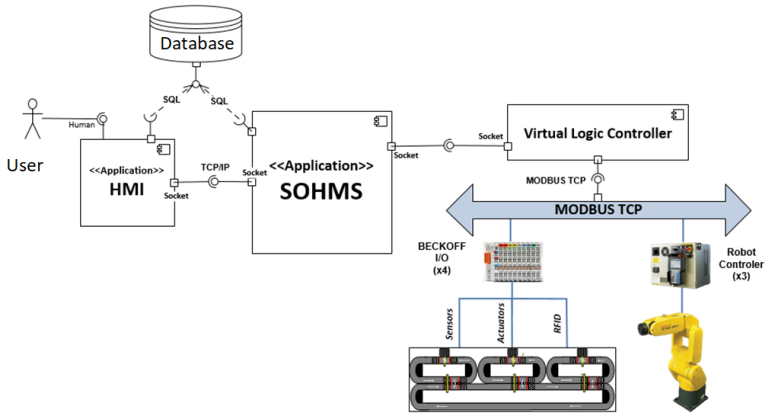

Fig. 1 Gamboa manufacturing software [Quintanilla, 2015]

current implementation consists of two applications which exchange informations through sockets. Both applications can be associated either to an emulation on Arena or to the real workshop via its logic controller [Gamboa Quintanilla et al., 2016]. Both are intimately coupled with the actual workshop contents so that any workshop modification implies a software review and maintenance. The programs includes 160 classes, 1240 methods and 14802 lines of code.

A study of the software architecture seen of Figure 1 pointed out several limits from a software quality point of view, including major trends such as maintenance and evolution, verification, adaptability and reconfiguration, etc.

- *Abstraction:* Except some limited UML documentation, the application is mainly java code which means that (i) the evolutions must be performed by experimented software developers, (ii) each workshop modification implies a serious development and verification work, (iii) during verification, business problem issues are melt with implementation issues (iv) the general mechanisms are not distinct from those specific to the workshop and lead to no simple reuse, no capitalization, no simple reconfiguration...
- *Maintenance and Evolution:* Agility is widespread now, users requires new requirements quickly. As the current manufacturing systems are highly diverse according to a mechanical or physical requirements, e.g., adding, modifying or deleting a mechanical component from the production workshop, any change can have a negative influence on the control software, which makes its maintenance and evolution a very delicate task.
- *Verification:* As the complexity of control software for manufacturing systems shall increase in the next decade, proving the correctness of such properties before deploying the application in a real production cell becomes a critical task. The current framework does not offer a model in which we can apply the verification techniques and ensure the requirements fulfillment such as : correctness, QoS, consistency, etc.
- *Adaptability and Reuse* Return on invest (ROI) applies not only when building workshops but also for the control software. The models and applications should be reused in different contexts. The current software architecture has not been implemented with an explicit and modular way that facilitates to reuse (parts of) the existing platform in different (rather similar) contexts.

The new vision intends to contribute to reduce these limitations.

3 Proposed approach

The proposal targets two goals: improve the quality of the software architecture (modularity, extendibility, variability) and improve the software construction process (abstraction, reuse, testability, reconfigurability).

3.1 Application Architecture

Instead of monolithic, nested and specific applications (left part of Figure 2) we target modular, independent and parametric applications (right part of Figure 2). The goal is to be as modular as possible in order to reduce the coupling between the different components, and reduce the specificity criteria of a production software to a precise context (layout, orders, resources, etc.).

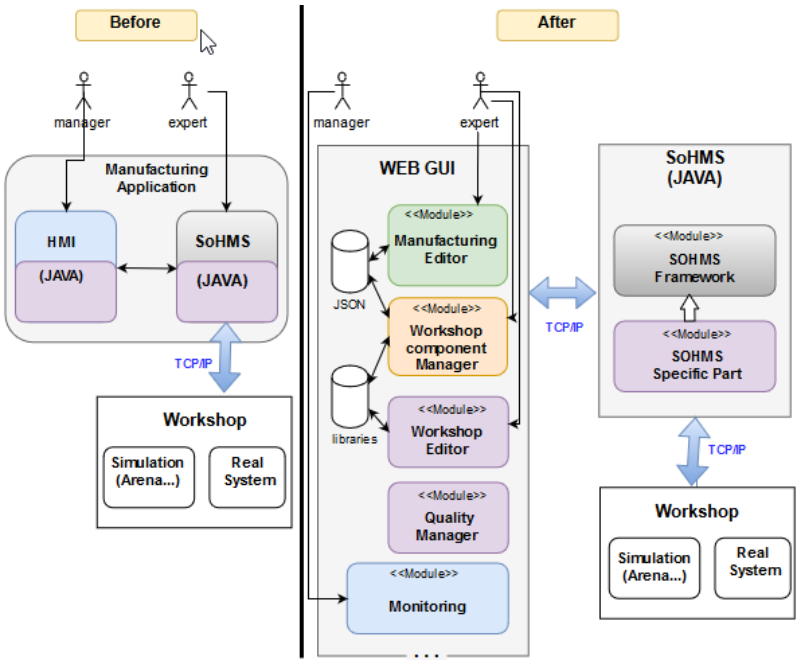

Fig. 2 SoHMS Software Architecture

The main and first issue is to separate the generic part, that stands for any kind of workshop, from the specific part which depends on the workshop resources and organization to put in place. The second issue is to separate the workshop features (resources and flows) from the manufacturing features (orders and products). The primer are rather static (even when considering reconfigurable manufacturing systems) while the latter are merely dynamic, time depending and subject to quality of service constraints. The GUI layer is separated in three applications: the system definition, configuration and monitoring. It is set up for constructing all interactions that an expert user can perform on a given manufacturing system, such as defining the various scenarios (Manufacturing editor), monitoring the running scenarios and define the layout on which each scenario will unfold. Technically, a web GUI layer enables not only to be independent from the operating systems and physical devices, but also a compliant property for connecting the applications to service-

oriented cloud or IoT and Scada platforms. The running SoHMS application drives the production system under the given constraints of the manufacturing process. The process engine, that includes the HMS definition, scheduling algorithms and properties, is generic while the orders, products and key indicators are specific. Finally the physical layer corresponds to different needs: emulation with Arena for example, management blackboards and of course the real workshop. The communication network between the layers should be compliant with different implementations (sockets, messages, shared memories...) according to the target infrastructure.

3.2 Software Construction Process

Basically, the existing SoHMS application (left part of Figure 3) consists of a collection of Java programs and workshop description which have been manually written and tested for a specific workshop case study.

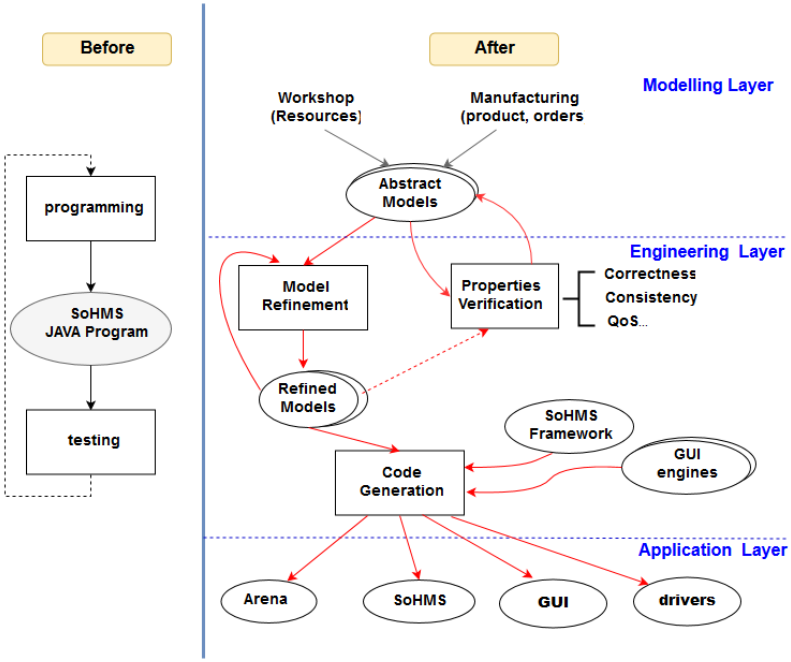

Fig. 3 Software Construction Architecture

The new vision (right part of Figure 3) installs a kind of software product line where people reason about the workshop to put in place before implementing it. HMS libraries are defined through components and services with related implementations, such that building manufacturing systems is rather an assembly and integra-

tion activity rather than a programming activity. Handling models enables to work with abstract concepts and to verify the expected properties without taking into account useless implementation details. These are qualified as platform independent models (PIM) in the MDA spirit while the current programs are really platform specific models (PSM). Figure 3 illustrates the proposed approach using three layers:

- *Modelling layer:* Working with abstract models enables to address the problem complexity and to formalize the system's structure, behaviour and requirements. Following the principle of "separation of concerns", building orthogonal models enforces consistency and reduces the software complexity. As an example, we have separated the development of workshop models that specify the structure of the production cells and the paths supporting the product transportation from the manufacturing models that specify the product recipes, the orders management and the orchestration of the process.
- *Engineering layer:* Model driven engineering means considering the model as a first class element in the development process. In this context, we aim to use the abstract models that we have defined in the modelling layer to apply : (i) formal verification, to get a high level of insurance about the correctness in early phase of the software development life cycle, (ii) models refinement, to automatically reduce the abstraction level until generate a concrete model that represents the application code, which increases the automation level in our approach. In addition we aim to construct standard and generic libraries and frameworks that can be integrated during the refinement process.
- *Implementation layer:* Various implementation targets but also various goals can be motivated. The implementation layer will support the automation process to generate the logic code for each concern designed in the modelling stage (GUI code, Emulation testbed, Manager blackboard and reporting, etc.).

Abstraction and separation of concerns are key issues for quality software design.

4 Experimentation

This section will briefly introduce the application context, an illustrative example and will include various references on previous works and experimentation. A very basic software production chain version is implemented to illustrate the process of Figure 3. In this section we will present the resulted manufacturing software and the different choices of technologies to implement it.

4.1 Case Study and Methodology

We could restart from the case study of [Quintanilla, 2015] and refactor the existing programs. However the reverse engineering and refactoring process of this

proof of concept (POC) application would take a long time because the specific and generic aspects of the solution are interleaved. This POC is still a resource for reverse-engineering but not the target system. So, the starting point for our case study was chosen as the Bench4Star [Trentesaux et al., 2013] manufacturing system. It is composed of seven workstations placed around a flexible transportation system. Workstations can have specific or concurrent capabilities. The production program consists of assembling components on a plate.

From a methodological point of view, we follow an agile perspective and will focus on the transportation issues for the first iterations. In parallel to this engineering stage, a reverse engineering process is applied to the current [Quintanilla, 2015] application in order to extract the main components and separate the generic and specific abstractions. This work provides fruitful components to build the generic SoHMS framework of Figure 3.

4.2 Workshop and Manufacturing Editor

The manufacturing editor, located in the GUI layer of Figure 3, aims to describe the workshop features (resources, layouts, products and orders). In its first version, a simple web-based textual editor has been developed to define the manufacturing scenarios to play later on (Figure 4).

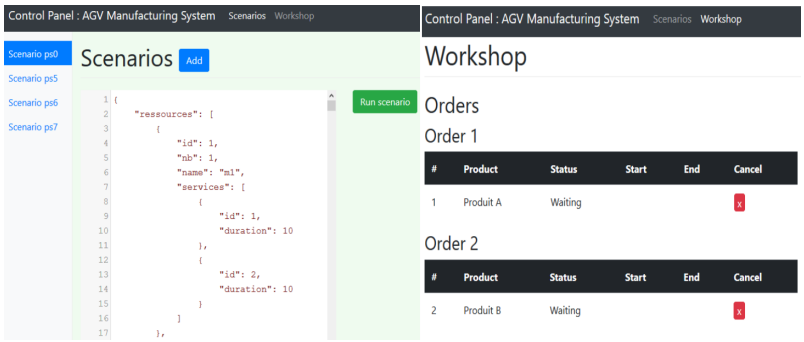

Fig. 4 Web GUI first implementation for CPPS control software

Scenarios are defined using `JSON` data format, a widespread data exchange language, easy to read or write for humans and simple to be parsed by different programming languages. The text editor is implemented using different web technologies such as `Vue.js, Bootstrap 4, HTML5, CodeMirror, WebScoket and PHP7`. Details on this editor can be found in [Carat et al., 2018]. The future versions will include GUI editors with menus and input forms equipped with drag and drop visual facilities to build workshops and manufacturing models.

4.3 Model Engineering (UML models)

For the modeling layer, the workshop and manufacturing models will be described using standard or domain specific languages (DSL) like UML, OCL, SysML, AADL, Kmelia as mentioned in [André and Cardin, 2018]. In the MDE spirit, the idea is to use a collection of languages, each of them enabling to capture a part of the system's semantics, say a point of view, that could serve as a support to reason about the system and to refine into implementable solutions.

Currently, we used UML diagrams to model elements of the manufacturing workshop and its behaviour. To find these elements, we got inspired by the application of [Quintanilla, 2015] and Bench4Star [Trentesaux et al., 2013]. For example, the class diagram of Figure 5 illustrates a simplified structural representation for the workshop (red part) and the production orders and scenarios (blue part).

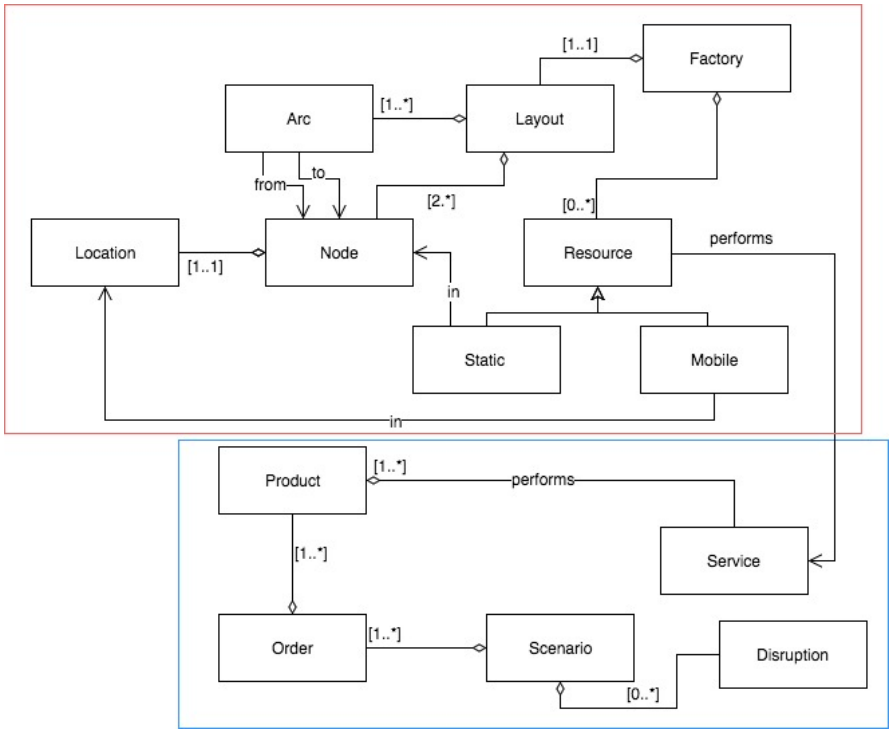

Fig. 5 SoHMS UML Models

To specify the behaviour of the system, we used UML sequence diagrams to express the interactions and statecharts to represent the component dynamic evolution. Details on the current modeling activity can be found in [Carat et al., 2018].

4.4 Model Transformation and Code Generation

The goal is to reduce the complexity of the development (programming and testing) by an iterative process made of simple processing task (modelling, verification, generation) *i.e.* small steps instead of big steps. Model transformation and code generation help to automate the refinement from high level abstract models to low level models (programs) that are specific to the target implementations. As an example, the JSON files of section 4.2 are used to generate the SoHMS specific elements. To generate the SoHMS code from the defined UML diagrams, we found that Papyrus[1], a graphical Modeling tool for UML with model transformation facilities, is a suitable tool to generate the corresponding code of our defined models basing on Model driven Engineering approach and using the model transformation concepts.

4.5 Toward Generic Frameworks and Libraries

We started a reverse-engineering activity that separate in [Quintanilla, 2015] application those classes that are generic, *i.e.* that are independent from the workshop to put in place, from those that are specific. The generic parts will be included in the SoHMS framework and the WebGUI framework: component libraries, HMS engine, default ordering policies, communication support (`WebSocket` technology), etc. The goal is to provide a generic and flexible communication support that can be extended to various platforms for developing such control systems.

5 Discussion

The use of MDE was already demonstrated on classical manufacturing control systems to improve productivity of over 50% compared to a hand-made code generation [Cuadrado et al., 2014], in flexibility and reconfigurability for SMEs [Masood et al., 2013] and in the application of products customization principles [Aleksic et al., 2012]. On the other hand, numerous issues in innovative CPPS control systems can be pointed out, among which:

- Lack of performance analysis and guarantees, due to the emergent behaviour targeted by the application of multi-agent concepts. It is then necessary to implement trustable validation and verifications techniques [Nastov et al., 2017] in order to guarantee the respect of the design constraints in the final code;
- Lack of a repeatable methodology, in order to stabilize the code generation process and increase the trust in the final behaviour of the system. This question is of major matter considering ethical questions, such as responsibility in case of major damages caused by the system [Trentesaux and Rault, 2017];

[1] `https://www.eclipse.org/papyrus/`

- Lack of stability of the underlying algorithms: in a Industry 4.0 context where service-oriented control systems are among the main objectives, the current developments need a major and constant rewriting of the code which prevent any reuse of codes or algorithms;
- Difficulty in migrating from and managing legacy environments: the control system's evolution is of a great concern when implementing innovative control systems to existing workshops. Indeed, the investment costs makes it quite impossible to change the resources accordingly with the software. Therefore, the software needs to adapt without rewriting code to a frequent reconfiguration of resources along the evolutions of the workshop layout and the customers' demand.

Considering this context, the objectives of this proposal are in two ways:

- Building a prototype software production chain in the context of next generation manufacturing control systems using MDE as a POC. By doing so, generic models and libraries are to be developed and constructed that will constitute a step beyond current literature in terms of genericity of CPPS control systems development.
- Develop the advantages of the application of MDE in this POC, in order to be able to demonstrate the Validation and Verification possibilities of such an approach and its benefits on the global process of control systems development: this objective is directly bound to the richness of the models used for generation.

6 Conclusion and Perspectives

Introducing the service orientation into CPPS control software was a first step to build scalable, modular and integrated control software. The next step is to introduce model driven engineering to build modular software systems that are easier to reconfigure, to maintain, but also to reuse parts into new other systems. We introduce the new construction systems and experimented parts to show its feasibility.

Future work will instrument the full software product line that implements our vision. At first, we will build the application engines *i.e.* the generic frameworks. Second, thanks to model transformation, we can generate the plugable software elements that are specific to the workshop to control. Third, we will write the code generators that enable to target different physical systems. Last, but not least, we will investigate the verification issues in order to implement trusty software systems.

References

[Aleksic et al., 2012] Aleksic, D. S., Jankovic, D. S., and Stoimenov, L. V. (2012). A case study on the object-oriented framework for modeling product families with the dominant variation of the topology in the one-of-a-kind production. *The International Journal of Advanced Manufacturing Technology*, 59(1-4):397–412.

[André and Cardin, 2018] André, P. and Cardin, O. (2018). *Trusted Services for Cyber Manufacturing Systems*, pages 359–370. Springer International Publishing, Cham.

[Bauer et al., 2017] Bauer, D., Stock, D., and Bauernhansl, T. (2017). Movement towards service-orientation and app-orientation in manufacturing {IT}. *Procedia {CIRP}*, 62:199 – 204. 10th {CIRP} Conference on Intelligent Computation in Manufacturing Engineering.

[Carat et al., 2018] Carat, A., Cherrueau, M., Courtoison, T., Girard, L., Grondin, M., Jain, E., Lemetayer, P., and Quémard, M. (2018). Contribution au logiciel de contrôle d'une flotte d'AGVs. Technical report, University of Nantes. (in french.

[Cuadrado et al., 2014] Cuadrado, J. S., Canovas Izquierdo, J. L., and Molina, J. G. (2014). Applying model-driven engineering in small software enterprises. *Science of Computer Programming*, 89, Part B:176–198.

[Gamboa Quintanilla et al., 2016] Gamboa Quintanilla, F., Cardin, O., L'Anton, A., and Castagna, P. (2016). Virtual Commissioning-Based Development and Implementation of a Service-Oriented Holonic Control for Retrofit Manufacturing Systems. In Borangiu, T., Trentesaux, D., Thomas, A., and McFarlane, D., editors, *Service Orientation in Holonic and Multi-Agent Manufacturing*, number 640 in Studies in Computational Intelligence, pages 233–242. Springer.

[Lee et al., 2016] Lee, J., Bagheri, B., and Jin, C. (2016). Introduction to cyber manufacturing. *Manufacturing Letters*, 8:11 – 15.

[Masood et al., 2013] Masood, T., Weston, R., and Rahimifard, A. (2013). A model-driven approach to enabling change capability in SMEs. *The International Journal of Advanced Manufacturing Technology*, 69(1-4):805–821.

[Meyer, 1988] Meyer, B. (1988). *Object-Oriented Software Construction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition.

[Monostori et al., 2016] Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, 65(2):621 – 641.

[Morariu et al., 2013] Morariu, C., Morariu, O., and Borangiu, T. (2013). Customer order management in service oriented holonic manufacturing. *Computers in Industry*, 64(8):1061 – 1072.

[Morgan and O'Donnell, 2015] Morgan, J. and O'Donnell, G. E. (2015). The cyber physical implementation of cloud manufactuirng monitoring systems. *Procedia CIRP*, 33:29 – 34. 9th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '14.

[Nastov et al., 2017] Nastov, B., Chapurlat, V., Pfister, F., and Dony, C. (2017). Mbse and v&v: a tool-equipped method for combining various v&v strategies. *IFAC-PapersOnLine*, 50(1):10538 – 10543. 20th IFAC World Congress.

[Quintanilla, 2015] Quintanilla, F. G. (2015). *Couplage des Architectures Holonique et Orientée-Services pour la Conception de Systèmes de Production Agiles*. PhD thesis, University of Nantes Angers-Le Mans-COMUE. PhD Thesis (in french.

[Quintanilla et al., 2016] Quintanilla, F. G., Cardin, O., L'anton, A., and Castagna, P. (2016). A modeling framework for manufacturing services in service-oriented holonic manufacturing systems. *Engineering Applications of Artificial Intelligence*, 55:26–36.

[Rodrigues et al., 2015] Rodrigues, N., Leitão, P., and Oliveira, E. C. (2015). Self-interested service-oriented agents based on trust and qos for dynamic reconfiguration. In Borangiu, T., Thomas, A., and Trentesaux, D., editors, *Service Orientation in Holonic and Multi-agent Manufacturing*, volume 594 of *Studies in Computational Intelligence*, pages 209–218. Springer.

[Trentesaux et al., 2013] Trentesaux, D., Pach, C., Bekrar, A., Sallez, Y., Berger, T., Bonte, T., Leitão, P., and Barbosa, J. (2013). Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice*, 21(9):1204–1225.

[Trentesaux and Rault, 2017] Trentesaux, D. and Rault, R. (2017). Designing ethical cyber-physical industrial systems. *IFAC-PapersOnLine*, 50(1):14934 – 14939. 20th IFAC World Congress.

[Wang et al., 2015] Wang, L., Törngren, M., and Onori, M. (2015). Current status and advancement of cyber-physical systems in manufacturing. *Journal of Manufacturing Systems*, 37:517 – 527.