Knowledge-based cyber-physical systems for assembly automation

Munir Merdan, Timon Hoebert, Erhard List and Wilfried Lepuschitz

Practical Robotics Institute Austria, Vienna, Austria

ABSTRACT

Current assembly systems are handling the increased requirements for mass customization with difficulties and need to be updated with new approaches and technologies. Cyber-Physical Systems (CPS) auto-configuration is regarded as an important asset towards automation components, which autonomously embed themselves into the system. In this context, knowledge-based technologies pave the way for highly flexible and reconfigurable CPS. This paper introduces and demonstrates a model-driven engineering approach for automatically configuring the control layer of a CPS based on knowledge representation of the environment and component capabilities. The approach encompasses a control architecture that is tested in two industrial use cases. The first case employs a configuration infrastructure for control software based on IEC 61499 to automatically configure the hardware-near control layer of a CPS within an assembly line. The second case is concerned with autonomously generating assembly plans, which are then transformed into actions that an industrial robot sequentially executes.

ARTICLE HISTORY

Received 19 December 2018
Accepted 9 May 2019

KEYWORDS
Semantics; ontology; knowledge-based reasoning; assembly line; industrial robot

1. Introduction

Production systems for the manufacturing of goods are often large complex systems containing numerous different subsystems. Assembly operations are considered being among the most intensive and costly processes in production lines. This is mainly due to the variability of the assembled parts and the complexity of the executed tasks, with the issue of rising complexity when the demand fluctuates in terms of volume and product types (Argyrou, Giannoulis, Papakostas, & Chryssolouris, 2016a). The assembly of manufactured goods accounts for over 50% of total production time and 20% of total production cost. In the automotive industry, 20–70% of the direct labor costs are spent on assembly. These statistics show the vital importance of assembly and indicate the prospective savings to be achieved by improving assembly approaches and technologies (ElMaraghy & ElMaraghy, 2016).

Currently, manufacturing systems are experiencing a paradigm shift with an ever-increasing variety of products combined with shorter product life cycles. The impact of this trend on them is that they have to produce highly individualized products with short delivery times and low manufacturing costs. In this context, the production steps
for assembling or machining products have to be adaptable based on the requirements of the product. Moreover, a more flexible logistic concept for the material flow among the machines and assembly stations is needed. Material flow systems (MFS), which cover the varying transportation routes related to the required product configuration and evolution can be very complex with the need to be constantly adapted (Regulin, Schütz, Aicher, & Vogel-Heuser, 2016). Further challenge relies on adopting assembly equipment that can switch between different production tasks ensuring process accuracy through auto-calibration. At the moment, the robotic technology seems as the only viable solution to cope with a paradigm shift. Thanks to the large number of assembly or manufacturing technologies that it can support, the robot-based production is seen as one of the crucial instruments to enable this transition (Michalos, Makris, & Chryssoulouris, 2015). Nevertheless, in order to cope with a new paradigm and meet a fully customized production with lot-size 1, more agile and intelligent robotic systems should be developed and deployed. The fast configuration of robots and correlated machinery as well as the acquisition and analysis of relevant process data are required. There is also a strong need for autonomous planning and execution of actions. Consequently, in order to handle the increased requirements for mass customization, assembly systems need to be updated with new technologies that focus on:

- reliable auto-configuration management for the assembly lines, and
- automated robotic system configuration functions, which enable a robot to autonomously plan its actions.

Cyber-Physical Systems (CPS) are regarded as a key technology for the development of future industrial systems by strong integration and tight coupling of computational and physical capabilities (Baheti & Gill, 2011; Kim & Kumar, 2012). Example applications for CPS can be found in the field of manufacturing control, energy systems, automotive and avionics systems, traffic control, medical systems, cooperative robotics and smart buildings (Facchinetti & Della Vedova, 2011; Saber & Kumar, 2010). It is widely agreed that an enormous economic and societal potential could be obtained by the application of CPS in fields such as the manufacturing domain (Lee, 2008).

As most CPS involve the cooperation of a high number of components (Kang, Kapitanova, & Son, 2012), explicitly programming the relationships between the system components and considering the quantity of interrelationships related to failures poses significant challenges (Wang & Lin, 2009). In this context, automatic configuration is proposed as a reasonable means for making a system scalable and robust in the presence of changes and for supporting dynamic adaptation (Kramer & Magee, 2007). Hence, technologies and methodologies are required for supporting the development process of large-scale CPS with physically distributed resources (Wolf, 2009). Model-driven engineering is a software development methodology which exploits domain models as a solution to handle the complexity of software development by raising the abstraction level and automating labor-intensive and error-prone tasks (Vyatkin, 2013). The approach has been promoted as a promising methodology for designing control applications of CPS (Balaji, Al Faruque, Dutt, Gupta, & Agarwal, 2015; Thramboulidis, Bochalis, & Bouloumpasis, 2017). Models allow domain specific software descriptions reflecting the heterogeneity of the developed system and may serve as primary development artifacts, which increases the
softwares comprehensibility and reusability (Ringert, Roth, Rumpe, & Wortmann, 2015). Besides, defined models are meant to be much more human-oriented than common code artefacts, which are naturally machine-oriented and software can be defined with concepts that are not necessarily dependent on the underlying platform or technology (Ciccozzi & Spalazzese, 2016). Furthermore, to develop a self-configurable CPS, a model-based approach may well be beneficial (Hehenberger et al., 2016). Self-configuration refers to the automatic deployment of a system configuration in response to changes in the environment. This system configuration is allocated to available hardware resources for instance by an autonomic service manager without any developer efforts (Dai, Dubinin, Christensen, Vyatkin, & Guan, 2017). Achieving a reliable auto-configuration management is an important challenge that has to be solved for enabling plug-and-play components, which autonomously embed themselves in the CPS without any special initialization procedure or elaborate human involvement (Reinhart et al., 2010). By using automatic configuration instead of manually adapting control software, engineering time and costs are reduced while increasing the softwares quality (Fischer, Vogel-Heuser, & Friedrich, 2015). This is especially of significance concerning the deployment and maintenance of nodes in large networks of small devices, e.g. sensor networks, as their manual configuration is rather impractical (Lanthaler, n.d.; Vicaire, Hoque, Xie, & Stankovic, 2012).

In order to manage the increasing complexity of manufacturing systems, knowledge representation concepts may be employed. The reactivity of future factory automation systems will be significantly increased by usage of knowledge-based systems on different control levels ranging from production order optimization to a reconfiguration of field level automation software (Legat, Lamparter, & Vogel-Heuser, 2013). In order to fulfill their tasks, the CPS components rely on a representation of their environment, e.g. an ontology describing the concepts and relations within the application domain (Gruber, 1993). Ontologies are considered as an intelligent way to manage knowledge, enabling the CPS components to agree on the meaning of common concepts they use in an open environment (González et al., 2006). Combining the concept of ontologies and explicit semantics makes the knowledge about the manufacturing system itself understandable to the control software and opens the way to automatic re-configurability of the control software in the production system (Negri, Fumagalli, Garetti, & Tanca, 2016). Besides, the positive effects of model-driven approaches can be further increased, if they are synthesized with ontological semantics and formal reasoning methods, as new forms of tool-support and assistance can be provided to software developers, system integrators and end users likewise (Zander, Ahmed, & Hua, 2016). Moreover, ontologies are reusable, extendible, and flexible, meaning that the population of data in ontological models is possible after model definition at runtime. This feature makes them even more powerful in the industrial domain because the system model changes continuously due to the large number of events occurring in manufacturing systems (Ramis Ferrer, Jarovyi, Lobov, Martinez Lastra, & Mohammed, 2016).

This paper introduces and demonstrates a generative programming approach for automatically configuring the hardware-near control layer of a CPS. A similar knowledge intensive control architecture is presented for the components of an assembly line (pallet transport system) as well as for the control of an industrial robot. The developed ontology in both cases covers the environment structure, characteristics, states and component interrelationships for enabling the CPS components to reason about these facts and
perform decisions. On the one hand, using their knowledge, the pallet transport system components (e.g. intersections) use a configuration infrastructure for control software based on the standard IEC 61499 iec1499:1 to automatically configure the hardware-near control layer of this distributed control system. Thereby, an executable LLC implementation is generated for each CPS component within the corresponding controller. On the other hand, ontologies enable the automated configuration of a robotics system. The CP component that is controlling the robot, generates the action sequences based on the semantic descriptions of the product and reason how a product should be assembled as well as in which manner parts should be moved from one position to another. Both cases are implemented and demonstrated on real equipment in a laboratory environment.

This paper is structured as follows. Related works and the problem statement are presented in Section 2. The pallet transport system use case and details of the control architecture as well as a resource ontology representing the target system and the actual configuration process are presented in Section 3. Section 4 presents in detail the industrial robot use case, the developed ontology, the used decision-making mechanism and the implementation of the approach. Finally, Section 6 concludes the paper with a summary and an outlook on further research issues.

2. Related works and problem statement

This section discusses related works and the resulting problem statement for this work on the basis of the two use cases.

2.1. Automated configuration

Much research in recent years regarding the automated generation of control code has been concerned with Programmable Logical Controllers (PLCs) based on the standard IEC 61131 iec1131 with the automatic generation of concrete software structures and implementations from the models. Current research in the field of model-driven engineering is mainly focused on developing new methods to support the development process of new software using modeling languages such as Unified Modeling Language (UML) or Systems Modeling Language (SysML) (Vogel-Heuser, Fischer, Feldmann, Ulewicz, & Rsch, 2017). Most approaches are based on UML, having the software structure and the behaviors automatically derived from the developed UML models as software artifacts into the target language (Tikhonov, Schtz, Ulewicz, & Vogel-Heuser, 2014). An approach based on UML for the automatic generation of field control strategies is presented by Legat, Schtz, and Vogel-Heuser (2014). Han and Park (2009) propose an object-oriented ladder logic development framework in which ladder logic code is generated automatically based on UML design results. Bonf, Fantuzzi, and Secchi (2013) present an object-oriented paradigm, which is applied by means of UML modeling techniques, to control software development for complex manufacturing systems with particular focus on case studies taken from the packaging industry. Vogel-Heuser, Witsch, and Katzke (2005) report an approach, which allows the generation of IEC 61131 code from a UML-model to import it automatically into soft-PLCs. Bergert, Diedrich, Kiefer, and Bar (2007) present a methodology to generate PLC software based on standardized digital process elements and existing planning information. Krzysztof (2005) describe a formal method for the automatic programming of
PLCs, which uses a subset of a UML-based state diagram model to define a correct control algorithm, and to implement the algorithm automatically using the ladder diagram language. Finally, Thramboulidis and Christoulakis (2016) present an approach based on a UML profile to automate the generation process of the IoT-compliant layer that is required for the cyber-physical component to be effectively integrated into the modern IoT manufacturing environment. However, several other researchers use SysML to generate PLC code. Gritzner and Greenyer (2018) report an approach for generating from scenario-based specifications written in Scenario Modeling Language, a compact and efficient Structured Text executable on PLCs. In Vogel-Heuser, Schtz, Frank, and Legat (2014) an approach for the model-driven development of automation software for manufacturing systems is presented. The approach is based on SysML and adapts the provided notations and meta-model to form a specialized language profile, SysML-AT, which covers (non-)functional requirements, corresponding software applications and properties of proprietary hardware components. Jamro Jamro (2014) presents an approach for the automatic generation of an IEC 61131–3 control software implementation using models created with the SysML graphical modeling language. It is also worth mentioning some other approaches for generating implementation code. Pavlovsky et al. (2018) propose in their work a technique based on principles of the declarative metaprogramming and template-based code generation for automatic generation of PLC software. Julius, Schrenberg, Schumacher, and Fay (2017) describe a systematic approach to automatically transform Grafcets into PLC code while retaining the hierarchical structures. Furthermore, Steinegger and Zotl Steinegger and Zotl (2012) use ontologies in their conceptual approach for generating PLC code by extracting data from various engineering models. In this context, Ferrer et al. (2015) present a framework that utilizes a knowledge-based approach to automate the component mapping in the code generation process. The dynamic elements are generated automatically from simulation models of manufacturing systems developed in 3D based virtual engineering tools. Besides, Priego, Schtz, Vogel-Heuser, and Marcos (2015) present an architecture that allows the automatic generation and update of IEC 61131 control code. It comprises several models for describing the components and operations of a plant as well as the work pieces and hardware components of PLCs. Plant and product model are parsed into a so-called hierarchical interconnected state space model with the implementation of ontological semantics.

A few concepts are also reported for automated code generation based on the standard IEC 61499 (IEC, 2005). Morán, Pérez, Orive, Estévez, and Marcos (2011) present a methodology and tool for automatically creating communication function blocks for distributed applications as a first step towards the automated composition of complete applications. The work of Garca, Castellanos, and Garca (2018) demonstrates the implementation of the IEC-61499 standard in low-cost devices by using UML diagrams as a software tool for modeling the function blocks. A knowledge-driven autonomic service management is proposed by Dai et al. (2017) to provide device level smart control of IEC 61499 compliant PLCs.

The European project MEDEIA was concerned with a more global approach for automatically linking the design and implementation phases. The cornerstone of this solution is a framework built around the concept of an Automation Component, which is an element that combines the descriptions about hardware, algorithms, interaction semantics and non-functional properties in a modular way (Colla, Leidi, & Semo, 2009).
Various methodologies and technologies for enabling automated configurability and increasing the flexibility within production automation systems are available. Model-driven approaches are still far from widespread industrial usage, since switching to such paradigms in industrial software development practice is complicated, and the required efforts are often very high (Tikhonov et al., 2014). The reasons are one side used modeling languages that are usually not familiar to the designated users (control practitioners) and model-driven approaches, which only allow performing modifications and revisions within the models to keep consistency of code and models (Julius et al., 2017). Furthermore, most of the above presented approaches focus on generating sequence-oriented PLC code in a monolithic manner. However, to deal with frequent configuration changes, as are likely to happen in flexible manufacturing systems, the logic control code needs to be easily modifiable and a new effective and intuitive method for logic code generation is needed (S. Lee, Ang, & Lee, 2006). Besides, the existing model-driven approaches either generate field level software, which is not applicable to IEC 61131–3, or generate control code with a fixed workflow (Bhrer, Legat, & Vogel-Heuser, 2015). Some of the approaches presented above tackle the latter challenge by basing their approach on IEC 61499. However, real implementations for automatically creating complete control applications based on this standard are as yet missing.

According to the paradigm of generative programming (Czarnecki & Eisenecker, 2000), the following objectives need to be tackled for achieving a so-called generative domain model and thus the basis for the automated configuration of the hardware-near control layer in a distributed CPS:

(1) Definition of the Problem space: contains application-oriented concepts and features which are mapped to the components of the solution space;

(2) Definition of the Solution space: contains the implementation components which should be flexibly combinable and reusable with a minimum of code redundancies;

(3) Definition of the Configuration knowledge: specifies the rules, dependencies, as well as illegal feature combinations for correctly mapping the features of the problem space to the components of the solution space.

Consequently, a generator then utilizes the generative domain model to create the implementation. To ensure a valid generation process, a specification check is performed using the constraints of the configuration knowledge. Finally, the executable implementation is a specific configuration of the implementation components of the solution space.

The components of the targeted manufacturing system need to be represented in the problem space as $PC = \{pc_0, ..., pc_n\}$ to enable their mapping onto the according software components $SC = \{sc_0, ..., sc_n\}$ of the solution space.

For defining the configuration knowledge, a rule base $R = \{r_1, ..., r_n\}$ has to be defined. On the one hand, each of the rules $r_i : \phi_i(x_i) \rightarrow \psi_i(y_i), i \in \{1, ..., k\}, k \leq n$, which are part of the sub-base $R_c$ for generating control code, consists of a condition function $\phi(\cdot)$ and an action function $\psi(\cdot)$ Steinegger and Zoitl (2012). On the other
hand, each rule \( r_j : \phi_j(x_j) \rightarrow \xi_j(y_j), j \in \{1, \ldots, l\}, l \leq n, k + l = n \) for the specification check in the sub-base \( R_s \) incorporates a constraint function \( \xi(\cdot) \) instead of the action function.

### 2.2. Industrial robotics

Currently used industrial robots are mostly statically placed in a cell, where they continuously repeat carefully pre-defined sequential actions. Moreover, industrial robots are very hard to program and require deep technical knowledge and work-time. In recent years, the industrial robot manufacturers started to address this issue by introducing different programming paradigms by demonstration methods, which requires less technical expertise and are more intuitive. However, these methods have still not found wide usage in the industry since they are not robust enough to generalize across a wide enough area such that they can be useful (Huckaby, Vassos, & Christensen, 2013).

To reach the needed level of flexibility and manage the increasing complexity of manufacturing systems, knowledge representation concepts may be employed. One of the key aspects needed to enable robot agility is to semantically represent the robot’s environment and its own capabilities in a way that it can reason about it and take actions based on what it has concluded (Kootbally, Kramer, Schlenoff, & Gupta, 2017). By building a system with appropriate knowledge and reasoning services it is possible to simplify the robot configuration sufficiently to meet the demands of flexibility while still getting a robust and efficient task execution (Stenmark & Malec, 2015). The usage of knowledge representation may allow to automate low-level decision-making and to provide a decision support for more complex tasks (Iarovyi et al., 2015).

The according knowledge representation and reasoning approaches have been successfully implemented in some industrial processes such as kitting (Crosby, Petrick, & Toscano, 2016; Kootbally et al., 2017) or in the European ROSETTA project (Patel, Hedelind, & Lozan-Villegas, 2012). This project focuses on the specification of how skills should be modeled for industrial assembly tasks. Also, the recent European Projects SMERobotics (Perzylo, Somani, Rickert, & Knoll, 2015) and SARAFun (Stenmark, Haage, & Topp, 2017) use partially the ROSETTA ontology aiming to increase the robustness of data exchanges as well as for the utilization of grounded concepts to aid specific phases of working with robots, such as configuration management and programming. Moreover, within the frame of the SMERobotics project, the BREP (Boundary Representation) ontology was developed. It was used for representing geometric entities in a semantic meaningful way and to improve a variety of applications ranging from shape-based object recognition to constraint-based robot task descriptions. Knowledge-enabled programming is also being used as a means to ease teaching a robot cell to assemble new products (Perzylo et al., 2015).

As can be seen, the exploitation of knowledge representation and reasoning methods in the industrial robot domain seems to be very promising. Considering that the assembly operations are regarded among the most intensive and costly processes in manufacturing lines (Argyrou, Giannoulis, Papakostas, & Chryssolouris, 2016b), this paper addresses this domain with a knowledge-driven automated robotic system framework able to cope with permanent product variations. Consequently, a two-layer-based knowledge intensive framework to control the automated robotic system is presented.
3. Use case: pallet transport system

The approach for an automated configuration of the control software has been implemented and evaluated in the first use case on a pallet transport system consisting of 45 conveyor belts with 32 intersections as well as 6 indexstations with grippers for holding pallets. Each component of this testbed (i.e. intersection, indexstation and conveyor) is represented by one CP component with monitoring and reconfiguration capabilities. This constitutes a CPS with intelligent interconnected components as it is required for the manufacturing domain (Cucinotta et al., 2009).

The following subsections introduce more details on the target system concerning the physical structure as well as the controlling Cyber-Physical component.

3.1. Testbed for distributed control

With the exception of 3 intersections all other intersections encompass either a switch with 2 different positions for routing pallets or a so-called three-way-switch with 3 different positions. Only the above mentioned 3 intersections incorporate no switch which means they cannot actively route pallets. Furthermore, the indexstations and intersections incorporate sensors for detecting pallets, blockers for stopping them and RFID modules for their identification. The subcomponents such as grippers, switches and blockers are equipped with sensors for verifying their position.

As presented in Figure 1, a set of 38 embedded controllers of the type CPX-CEC-C1 by Festo is employed to control the indexstations and intersections as well as the conveyors. Each Festo CPX hosts an Xscale-PXA255 agile Intel microprocessor with 400 MHz, 28 MB Flash and 24 MB RAM and several I/O modules (digital inputs, digital outputs and valves) located in the controllers' slots which are connected to the sensors and actuators. A Linux operating system is employed on this type of controller. Each

Figure 1. Picture of the pallet transport system.
RFID module is connected via a serial interface with a small embedded controller of the type Digi Connect ME. In total, 118 controllers are utilized for hosting the hardware-near control software and thereby constitute this distributed CPS.

### 3.2. Cyber-physical system component architecture

The architecture of the CPS component represents the core concept of the CPS for controlling the testbed. Each component incorporates the controlled physical part, which can be regarded as its embodiment and a software part which acts as the component’s intelligence. Segmenting a system into entities each controlling one component with local goals offers flexibility and enables therefore the optimization of the whole manufacturing process (Metzger & Polaków, 2011). Such an architecture was employed successfully for achieving a fault-tolerant system (Merdan, Vallée, Lepuschitz, & Zoitl, 2011; Vallée, Merdan, Lepuschitz, & Koppensteiner, 2011).

Considering the requirements of the manufacturing environment for real-time tasks, it is widely agreed to split manufacturing control into two layers (see Figure 2): the high level control (HLC) and the hardware-near low level control (LLC) (Christensen, 2003). In the presented approach, the HLC of each CPS component runs on a PC while the LLC is executed on the embedded controllers. Hence, the two control layers of a component communicate over the laboratory’s network.

![Figure 2. Architecture of a cyber-physical component – only the HLC elements required for the automated configuration are depicted.](image-url)
3.2.1. High level control

The HLC is responsible for controlling the behavior of the CPS component in coordination with the neighboring entities. It incorporates a World Model repository for storing a symbolic representation of the surrounding world. To reason about the states of the world as well as about inconsistencies between the world representation and information received from the environment, the HLC can query the contents of the world model before initiating actions. Most likely, the World Model is updated after recognizing changed world conditions or after performing actions. The World Model in this architecture incorporates two parts: the Semantic Model and the Activity Model.

The Semantic Model holds knowledge about the component’s own characteristics and its relation to the surrounding environment (i.e. other entities). The basic knowledge structure is constituted by an ontology. In contrast to CPS with centralized ontologies (González et al., 2006), each CPS component relies on its own ontology which is derived from a general system ontology. The vocabulary defined by the ontology is then used to express knowledge about the current state of the world in a set of facts. However, only meaningful aspects are represented instead of representing the complete world around the component.

The Activity Model contains an abstract representation of the actions, tasks and goals of the CPS component that can either be observed in the system or are expected to take place. As the approach presented in this paper is not focused on the component activities during operation, the well-disposed reader might find more detailed information concerning the concept of the Activity Model in (Merdan et al., 2011; Vallée et al., 2009).

The Decision Making as used in this approach for automated configuration (see Section 3.3) takes knowledge from the Semantic Model as Problem Space description. The Solution Space is composed of information about the LLC components that can be used for creating the LLC. The Configuration Knowledge represents the rule base that is required for actually performing the automated configuration by the generator.

The HLC is implemented in Java, and the Jade platform Jade is used as infrastructure for Cyber-Physical component management and inter-component communication (Jade 2018).

3.2.2. Low level control

The hardware-near LLC has access to the subjacent physical components, i.e. sensor and actuators, and provides a set of reactive behaviors. The industrial standard IEC 61,131 IECIEC (1993), which specifies languages for Programmable Logical Controllers (PLCs), is widely spread in the manufacturing domain. However, PLCs based on IEC 61,131 represent a control architecture with a centralized approach which limits the system’s modularity and reconfigurability (Brennan, 2007; Cucinotta et al., 2009). This centralized approach commonly leads to applications based on the usage of global variables which effectively link seemingly decoupled program parts via a hidden interface. Therefore, reconfiguring such program parts can result in major consequences in other parts of the application. Moreover, due to the data-driven approach of IEC 61,131 and its cyclic execution model, the exact execution order is not specified explicitly. Hence, for the reconfiguration of current IEC 61,131–3 systems the applications are changed at once on the whole and not gradually as it would be preferable for a distributed control system with modular components (Lepuschitz, Zoitl, Vallée, &
Merdan, 2011). A different control paradigm supporting logical and spatial software distribution and reconfiguration such as IEC 61,499 is therefore recommended for flexible distributed CPS in the manufacturing domain (Metzger & Polaków, 2011; Vyatkin, 2011).

This perception goes along with the two key issues identified concerning the development of the LLC for the approach presented in this paper: an increased (automatic) element reuse as well as the dynamic LLC configuration and reconfiguration in order to adapt quickly to changing production requirements. The acceptance of the distributed concept for industrial applications could be increased significantly with an easy, fast, transparent and reusable integration of physical automation components (Leitao, 2009). A modular strong component-oriented architecture with control software components interacting via defined interfaces is therefore the enabling technology (Zoitl, 2009).

The standard IEC 61,499 introduces a generic architecture and guidelines for the usage of function blocks in distributed industrial control systems. A function block (FB) represents a functional unit of software containing data and algorithms as well as internal variables to constitute an internal state (distinguishing it therefore from a pure function). In contrast to IEC 61,131, the FBs of IEC 61,499 are event-driven, which increases their portability as well as reusability and the controllability of a control application, as the execution sequence is explicitly determined (Lepuschitz et al., 2011). References to several works of this standard’s aptitude towards reconfiguration are reported in (Vyatkin, 2011). Hence, the standard IEC 61,499 is identified as the most suitable concept for an automated control software configuration of a flexible distributed CPS.

The typical LLC application for each component of the pallet transport system contains only a small number of FBs, i.e. eleven FBs in the case of a diverter’s LLC (see Figure 3). Ten of these FBs are responsible for controlling the subcomponents of a component such as sensors and blockers with one FB per subcomponent. The three FBs FB_SensorDevice handle the functionality of the three pallet sensors and the three FBs FB_BlockerDevice are responsible for the three blockers. FB_Switch3wayDevice controls the three-way-switch and is substituted by another FB in the case of intersections incorporating a switch with only 2 positions. Three FBs FB_RFID_Device are used for receiving data from the RFID readers at the adjacent conveyors. In contrast to the FBs controlling the subcomponents, higher functionalities such as the communication with the HLC and the routing of pallets are provided by FB_Diverter. For carrying out its algorithms, FB_Diverter requires access to the pallet sensor, blocker device and the RFID device located at the incoming conveyor, as well as to the pallet sensors located at the two outgoing conveyors. By using the adapter concept for the connections between the FBs, only six connections are necessary. On the one hand this allows for an easier comprehension by humans, which is an advantage regarding debugging. On the other hand a very decoupled application design is achieved, where components can be exchanged within only a few steps.

3.2.3. Ontology of the control system

For the automated configuration of the control software, an ontological representation is required, which contains sufficient information concerning the structure of the target system. A few approaches mentioned in Section 2 employ ontologies as a basis for the generation of control code.
The software configuration process presented in this paper relies on an appropriate resource ontology regarding the physical components. Several ontologies have been developed and presented for specific purposes concerning the manufacturing domain. A taxonomy of manufacturing components with physical resources (machines, tools, etc.) and concepts for production orders, operations and disturbances is described by the ADACOR ontology (Leitao, 2004). The MASON ontology focuses on likewise aspects with machines as well as tools considered as resources and raw material as

Figure 3. Desired LLC application for an automation agent acting as a diverter (parameter values of the FBs are not shown in this figure).
well as other common helper concepts regarded as entities. Operations describe the process and cover the performed procedures (Lemaignan, Siadat, Dantan, & Semenenko, 2006). An equipment module ontology with focus on the functions and behaviors of equipment entities and their connections is presented in (Lohse, Hirani, & Ratchev, 2006). Other research has been concerned with merging ontologies for mechatronic devices and automation reference models covering both hardware and software features (López Orozco & Lastra, 2006).

However, for an automated configuration of the control software the above mentioned ontologies are only partially suitable as they do not take the actual control infrastructure into account. But controllers such as PLCs and embedded industrial controllers should be included to facilitate such an integration process (Lastra & Delamer, 2006). An ontology comprising also the concept of controllers for reconfiguration tasks performed by a global reconfiguration agent is described in (Alsaifi & Vyatkin, 2010). Another resource ontology for the manufacturing domain, which includes controllers and their related concepts and is partially based on the concepts of the standard IEC 61,512–1 Batch Control, Part 1: Models and Terminology (IEC, 1997), respectively ISA S88.01, is presented in (Lepuschitz, Zoitl, & Merdan, 2011).

The focus of the work described in this paper is not to develop an extensive ontology for the manufacturing domain but to demonstrate its suitability for the automated configuration of the CPS. Hence, the latter two ontologies mentioned above were taken into consideration for the development of the hardware-related part of the system ontology (see Figure 4, Hardware). According to IEC 61,512, basic components such as blockers and switches, which are regarded as subcomponents in this approach and form the smallest entities realizing some basic automation, are referred to as control modules and are therefore represented as specific subclasses of the class ControlModule. Control modules can be aggregated to so-called equipment modules, which is denoted in the ontology with the relation hasModule between the classes EquipmentModule and ControlModule. In the testbed, components such as the different types of intersections, which are composed of

![Figure 4. Overview of the system ontology.](image)
sensors, blockers, RFID modules and most likely a switch, are subclasses of the class EquipmentModule. Both module types are furthermore subclasses of the class Module. Conveyors are not part of specific equipment modules and therefore their class is derived directly from the class Module. However, they are linked with the equipment modules to form the topology of the pallet transport system. Each module is controlled by a controller and the employed types of controllers are derived as subclasses from the class Controller.

Further semantic extensions in the form of a software-related part were added for the wiring of the LLC application (see Figure 4, Software). The class FunctionBlock refers in principle to software components of any type of component-oriented control software and does not necessarily refer to an IEC 61,499 FB. This general concept is of importance in the case this approach is adopted for configuring a different type of LLC. Each module type is represented in the control software by a specific type of FB. The class Parameter refers to the parameters of the software components such as the specific pin number at the controller the module is connected to. The nodes of a module’s FB represent the connection points to the software components of other modules within the same LLC application. In order to achieve a valid executable LLC, all required nodes of the LLC’s module FBs need to have connections to provided nodes which fulfill particular restrictions. Generally, the providers are linked with their FB’s parameters and specific values are expected by the requirements of other FBs, which are described in the requirement’s restrictions.

This ontology represents the problem space of the automated configuration and its contents are stored in an according XML format. Using XML for the design artefacts of the ontology goes along with advantages such as its platform independence as well as a self-explanatory representation (Vyatkin, 2011). The classes of the resource ontology with their relations are stored in a module specification file. The instances are stored in a corresponding instance specification file.

### 3.3. Configuration of the LLC

At system startup, the HLC for each equipment module is created automatically by a central unit of the CPS. Each component’s HLC then acts as a generator which has to configure its LLC accordingly. For this purpose the HLC relies on the module and instance specification files, which represent the resource ontology of the problem space, to determine the components physical parts (sensors, blockers, etc.) and how they are connected with the corresponding controller (i.e. specific pin of an I/O module at a Festo CPX). Furthermore, the FB types are provided in XML format as the components of the solution space. According to the component-oriented design of the LLC (see Section 3.2.2) each FB incorporates either the functionality of one specific subcomponent (e.g. FB_BlockerDevice) or the higher LLC functionalities of a whole component which includes the communication with the HLC or the routing of pallets (e.g. FB_Diverter). Finally, a set of constraints forming the integrity rules for the specification check and generation rules for the actual LLC configuration are embedded in a set of Java methods that can be used by the CPS components. The HLC then sends a sequence of commands to the controller, which hosts its according LLC, for generating the executable implementation. Figure 5 depicts the workflow of the configuration process.
3.3.1 Configuration services

The configuration process starts with the creation of a resource in the target device followed by the creation, parameterization and wiring of the corresponding FBs. A resource is created if there is at least one instance defined in the instance specification. For all instances, the FBs are then created and the according parameters are written. Afterwards, each Requirement instance x is connected with a corresponding Provider instance y which meets the Restrictions defined in the module specification. Finally, the FB network is initialized by issuing a start command.

Configuring the LLC requires a sequence of services for changing the structure of the control application (i.e. creation and deletion of FBs and connections as well as the modification of parameters) as well as execution control services for changing the execution state of FBs. A real-time capable infrastructure for reconfiguring IEC 61,499 applications of distributed embedded control systems is presented in (Zoitl, Lepuschitz, Merdan, & Vallée, 2010). In this approach the reconfiguration services are realized with commands, respectively requests, using an XML format as defined in the standard IEC 61,499, which are sent to the device management in the corresponding controller. Each request is followed by a response of the device management before the next request is issued. This infrastructure is suitable for the LLC configuration process presented in this paper.

As the HLC of the CPS component is realized in Java, the configuration services are embedded in a set of Java methods which compose the according XML commands. Figure 6 shows a part of the command sequence starting with the resource creation (command 0 in the figure) and ending with the start of the resource for configuring the LLC of a diverter in the pallet transport system. The shown commands between the resource creation and its start, for creating FBs, writing their parameters and creating the connections, are just a selection as the other commands for the configuration process are structured likewise.
3.3.2. Performance results

As already mentioned, the dynamic configuration of an industrial control system is a key technology for achieving flexible production systems. Depending on the timing constraints of the manufacturing environment and to avoid an extensive down-time of the system components, such a process has to be carried out in a specific time frame.

Measurements of a series of configuration runs with one CP component show that this process requires approximately 1 second (see Figure 7). Configuring the LLC of this component acting as a diverter in the pallet transport system requires its HLC to send a sequence of 77 commands over the network to the corresponding Festo CPX controller. Consequently, the execution of each command takes place within 13 ms. These results are in line with timing results for a reconfiguration process performed on the

| No | Type of action          | XML Command                                                                 |
|----|-------------------------|-----------------------------------------------------------------------------|
| 0  | Creating resource      | <Request ID="0" Action="CREATE"> <FB Name="EMB_RES" Type="EMB_RES"/> <Request> |
| 1  | Creating new FB        | <Request ID="1" Action="CREATE"> <FB Name="Sensor_L" Type="FB_SensorDevice"/> <Request> |
|    |                        |                                                                             |
| 31 | Setting parameter      | <Request ID="31" Action="WRITE"> <Connection Destination="Sensor_L.SlotNrS" Source="1"/> <Request> |
| 32 | Setting parameter      | <Request ID="32" Action="WRITE"> <Connection Destination="Sensor_L.BitNrs" Source="1"/> <Request> |
| 71 | Creating new connection| <Request ID="71" Action="CREATE"> <Connection Destination="Diverter.Sensor_In_LR"> Source="Sensor_L.Adapter_Sensor" /> <Request> |
| 76 | Starting LLC           | <Request ID="76" Action="START">                                          |

Figure 6. Issued commands for configuring the automation agent’s LLC.

Figure 7. Execution time of the configuration of the LLC in 1 controller (for 10 different configuration runs).
basis of the same IEC 61,499 reconfiguration infrastructure for the FORTE (Lepuschitz, Vallée, Zoitl, & Merdan, 2010; Lepuschitz et al., 2011). Even though most runs require almost exactly 1 second, run #3 required about 300 ms more while run #4 required about 300 ms less. These significant differences might be caused by network traffic, which means that a configuration process could be even faster if the HLC resides directly within the same controller as the LLC.

Due to the distributed nature of the target system and the controlling CPS, the configuration of the CPS components’ LLC can be performed in parallel threads. This proves to be a significant advantage compared to a centralized system with one PLC hosting the hardware-near application of the complete pallet transport system. As the size of a monolithic PLC program grows with the amount of controlled components, its compilation and download time also increases. On the contrary, using parallel threads for configuring several controllers with the approach presented in this paper keeps the configuration time at roughly 1 second as measurements show when 5 CPS components configure their LLCs in parallel threads (see Figure 8). Further timing measurements with 10 or 15 controllers reveal a total configuration time of approximately 1 second as well. Regardless of the number of configured LLCs and thereby configured controllers, the configuration time does not change significantly. On the contrary, in the case of a centralized system with a monolithic program for the complete pallet transport system running on one controller, the compilation and download time of this program would be considerably longer. This clearly shows an advantage of distribution in a CPS compared to a centralized approach.

However, likewise to compiling a program for a PLC, the configuration time increases with the number of required commands for each distinct controller. In this context, the component-oriented design of the LLC as used for this implementation is also evaluated in comparison with a less decoupled application design not based on adapter connections between the FBs previously used for controlling the pallet transport system. For configuring this kind of LLC, the CP component did not rely on the approach presented in this paper but on pre-defined XML files containing a description of the required FB network. Even though

![Figure 8](image-url)
distributed on several controllers, the rather monolithic design of these previously used LLC applications on each controller impedes an easy exchange of FBs and thereby the dynamic reconfiguration of the control software. Therefore, the application for an CP component’s LLC has to incorporate the functionality for all expected system states following thereby a contingencies approach of reconfiguration (Brennan, Fletcher, & Norrie, 2002). Hence, in the case of a CPS component controlling an intersection starting as a diverter, an additional functionality has to be provided in the FB network already at system startup if an adjacent conveyor’s direction would be changed due to a required system reconfiguration. Creating and wiring these additional FBs as well as using no adapter connections between the FBs in general results in a much longer configuration process with 500 commands taking a configuration time of about 7.5 seconds (see Figure 9(a)). Hence, using the component-oriented design for the LLC in the Festo CPX reduces the configuration time by 87% and keeps it thereby within acceptable boundaries. Likewise measurements concerning the configuration of the small FB network in the Digi Connect ME controllers, providing access to the RFID modules, also show a reduction of the configuration time by 88% due to the component-oriented design (see Figure 9(b)).

4. Use case: industrial robot

The approach for an automated configuration and programming of the industrial robot is demonstrated in a laboratory environment with an industrial pilot test case that focuses on the assembling of different THT-Devices (relay, capacitor, screw terminal, and potentiometer) on a PCB. The challenge of this task is to switch accordingly between the different PCB layouts and assemble correct THT-Devices using the appropriate tool.

The presented architecture in Figure 10, same as for the first use case, is based on an existing concept, where the control of an individual manufacturing component was separated into two levels: high-level control (HLC) and low-level control (LLC).
A clear repartition of roles and algorithms between these layers is important for the robustness and efficiency of a specific manufacturing component, i.e. in our case the industrial robot.

The HLC in this use case is responsible for the management of the robotics systems, integrating planning, scheduling and coordination functions in this layer. Its functions encompass mostly the calculation of larger amounts of data and are therefore not performed under real-time constraints. It consists of the World Model and Decision-Making components. The latter one is designated to receive orders from outside and identify the desired aim (product) in the World Model. The Problem Generator collects necessary information from the World Model to fulfill an assignment. Based on this information the Planner automatically determines action sequences required to produce the given product. It uses this information to select tools that have the required skills to fulfill the actions and also plans the execution order of these operations in order to achieve minimal production time. However, the selected plan is strongly related to the current state of each component in the World Model as well as the present state of the robot. Because of this reason, the Planner permanently examines the World Model to obtain the current state of the entire robot system and select the plan with a list of achievable actions. These actions are then sent to the Plan Executor, which translates these actions into the language understandable for the related robot controller.

The World Model plays a central role in the architecture of this robotics system. It contains knowledge about all aspects of the robotics systems and the related production environments wherein this system is used. Consequently it incorporates information about robot capabilities (skills), used tools, production processes, related external

![Control architecture for industrial robots.](image)
components as well as concrete positions (3D coordinates) of all relevant system components. The World Model consists of two parts. The Semantic Model contains the semantic representation of the robotics system characteristics and all relevant entities in the production environment as well as relations between them. The Activity Model holds knowledge about the current state of the world containing all instances, current events, and processes occurring in the world in which the robot is currently participating. It is permanently updated with processed sensor information as well as through the communication of the robot system with external production environment entities. Generally, the Semantic Model contains classes (concepts), attributes as well as relations and all this information including instances are available for the Planner through the Activity Model. The World Model repository provides facilities for updating and querying the World Model.

The low-level control (LLC) is a robot controller responsible for managing the physical component by using a set of primitive behaviors as well as required sensing operations, which are essential for the robot to work. The primitive behaviors are characterized through an execution single operation action by the robot system. Examples of primitives include configuration and calibrating the position of the robot, opening and closing the gripper, approaching the object as well as moving the object from one to another approaching position. These primitives are simple, atomic robot actions that can be combined to form more complex behavior, which is often called a robot skill (Stenmark, 2011). For example, the place object skill is a combination of the following primitive actions: move the arm, approach, release and retract. Each skill has a related set of preconditions and a forecast of the outcome after its completion in the real world, which are then used to examine system states before and after execution. These can be later verified by checking the World Model or through ad-hoc sensing operations (Pedersen, 2014).

Finally, the Plan Executor application, which is a part of decision-making component translates each action from the plan file into a series of Canonical Robot Command Language (CRCL) commands and sends each to the Robot Controller (Proctor et al., 2016). CRCL provides message content and the syntax for a low-level robot command-status protocol.

4.1. Ontology

Ontologies are necessary to enable a knowledge-driven configuration of a robotics system in an extensible manner. A big challenge lies in finding a suitable abstraction level for describing the capabilities of the system components. To be adequately used for system configuration, the ontology needs to represent and map an accurate model of the robot environment, including tools but also other machines, devices, interfaces, and services. Considering the significance of existing research concepts and ontologies, parts of the existing ontologies were integrated into the semantic system model of this work and merged with new concepts and class hierarchies. As a basis, the Rosetta ontology was used, in which the robotic devices and particularly skills have been put in the center or concept integrating also the BREP ontology, which represents geometric entities in a semantic meaningful way. The aim is to semantically specify all relevant concepts of a robotic system (e.g. capabilities/skills, properties, constraints) as well as the related manufacturing environment (product, storage, sensors, feeder, etc.), which is needed to
reach specific goals. Considering that the focus is on the assembly domain, an assembly ontology was integrated that has its roots in the ontology-based assembly model presented in (K.-Y. Kim, Manley, & Yang, 2006). The assembly ontology was initially developed in (Merdan, 2009) and extended in (List, 2015). The ontology generally integrates knowledge needed by the robot to autonomously reach specific goals (e.g. assemble a product). The Part(s) being manufactured have a central role in the ontology. They are specified by a set of properties (size, shape, color, etc.) and constraints e.g. grasping requirements. The defined Assembly Step (see Figure 11) uses these parts and their interrelations to make an Assembly. At the same time, each Assembly integrates further information, such as geometrical relations between the parts or what type of joining operation will be applied (e.g. stitching, screwing, welding). The kind of assembly tree, which arise when several Assemblies (or at least one Assembly and the Part) are connected tougher, specifies also the order of the assembly process and related operations, where child assemblies have to be joined first.

The integrated relations needsPredecessor and isFollowedBy enable the specified order relationship between the two corresponding Parts or Assemblies. In this context, Assembly Constraints define the orientation of the components in the assembly, but also restrictions of the relative motions between object features. Robot arm motion constraints are determined by the motion planning through the production environment encompassing position, velocity and motion type. There are also Operation Constraints, limiting how a specific operation should be executed including time and space restrictions as well as when and how to use specific tools. Our semantic model is conceptually defined using the Web Ontology Language (OWL) standard and is automatically imported into our activity model and loaded into a graph database.

4.2. Decision-making

The robot has to reason about how a product should be assembled as well as in which manner parts should be moved from one position to another. The decision-making mechanism (Planner) acts as a link between the semantic model of the production environment and the available robot system capabilities (skills). In this context, the Decision-Making component uses the ontology-based product model to reason about relations between the assembled parts as well as to connect assembly processes and needed assembly equipment. Reasoning and plan generation is performed in a bottom-up manner starting from the top of the assembly tree where the final product is specified and follows

![Figure 11. Product model and semantic representation (reduced).](image-url)
the inverse order of the assembly plan until the bottom of the tree is reached consisting only of the first parts that have to be connected at the beginning of the assembly process. An assembly is a nonempty subset of parts that either has only one component or is such that every part has at least one surface contact with another part in the subset (Rabemanantsoa, 1993). The relationship between parts within an assembly defines operations that have to be done to connect these parts and represents how these subassemblies should be put together to complete the product. An operation again leads to the required skill that specific manufacturing resource needs to have in order to change a state of the part. Based on this information and in order to reach a desired goal state from the initial state, the Planner discovers a concrete sequence of assembly steps and related operations, where each operation is represented as a set of primitive motions and actions coded on the robot controller.

To generate operation plans, the Planning Domain Definition Language (PDDL) (Mcdermott et al., 1998) is used for decision-making. It is a standard language widely applied in the academic planning community. The PDDL input format consists of two files that specify the domain and the problem. A domain file declares the predicates, functions, and types of objects available on the world and the actions available. In this context, each property of each class in the Semantic Model can be transformed to a PDDL predicate for describing the states and actions. For example, part locations are specified by the predicates hasLocation pointing on the specific box where the part is located. Another example is the free predicate, which specifies if the concrete gripper is holding something at the particular moment. Actions are the means of how the world state is changed. The statement includes parameters, preconditions, and effects. The preconditions dictate items that must be initially valid for the action to be legal. The effect equation dictates the changes in the world that will occur due to the execution of the action. For example, the following statement is an action that specifies the action of a robot to pick a part from a specific location:

```
(:action pick
 :parameters (?obj – part ?box – Position ?gripper – gripper)
 :precondition (and (hasLocation ?obj ?box) hasPosition ?box)
          (free ?gripper))
 :effect (and (carry ?obj ?gripper)
            (not (hasLocation ?obj ?box))
            (not (free ?gripper))))
```

The problem file specifies information about the specific instance of the given problem. It describes the state of the world at the start and the goal state to reach. For example, the: init statement specifies the initial position of the relays RY211005_GRAU and RY211005_ORANGE, which have a location in the BOX_RELAIS_GRAU and BOX_RELAIS_ORANGE boxes respectively as well as clamp ML252-10-D-LC that has the location at the box BOX_STECKER. The goal statement specifies the goal destination of these three parts. In order to move these parts, the vacuumGripper should be used as a tool.

```
(:init
 (hasPosition Initial)
 (free vacuumGripper)
 (hasLocation RY211005_GRAU BOX_RELAIS_GRAU))
```
With these two specifications the PDDL planner can be set to search and find a solution, that is in effect an appropriate sequence instance for this manufacturing domain/problem combination.

(move home box_relais_grau)
(pick ry211005_grau box_relais_grau vacuumGripper)
(move box_relais_grau pcb_tele)
(drop ry211005_grau pcb_tele vacuumGripper)
(move pcb_tele box_relais_orange)
(pick ry211005_orange box_relais_orange vacuumGripper)
(move box_relais_orange pcb_tele)
(drop ry211005_orange pcb_tele vacuumGripper)
(move pcb_tele box_stecker)
(pick ml252-10-d-lc box_stecker vacuumGripper)
(move box_stecker pcb_tele)
(drop ml252-10-d-lc pcb_tele vacuumGripper)

The Executor application frequently follows the execution status of each CRCL command, ensuring that the conditions and effects of the PDDL commands are met as the statements are processed.

4.3. Implementation

The presented approach is demonstrated in a laboratory environment with an industrial pilot test case that focuses on the assembling of different THT-Devices (relay, capacitor, screw terminal, and potentiometer) on a PCB. One of the layouts is presented in Figure 11 and parts with a red dot are relevant for the automated assembly concepts. In this scenario, THT-Devices for an assembly line are placed in boxes at the workstation. Each box contains different parts, which have an exact position in the box. In the automated pilot case, the KUKA robot KR 6 R900 sixx performs a series of pick-and-place operations in assembling the final PCB board, as presented in Figure 12. These operations include:

- Pick the empty PCB board and place it on the work table.
- Pick relevant THT-Devices and place it on the board.
- Assembled PCB board is picked and placed in the storage area.

The developed semantic models contain representations about all relevant concepts used in the pilot test case.

According to the demand for a central database server in the industrial domain for supporting the high-performance integration of multiple homogeneous data sources,
for example to query geometric data for a vision system, the Neo4j named property graph engine is used. The LITMUS benchmark (Thakkar, Keswani, Dubey, Lehmann, & Auer, 2017) compares multiple database engines suitable for semantic web data (RDF-Triple/Quad-Stores and Graph databases) and rates Neo4j as a high-performance system. It is open-source and has been adopted by the industry.

The conversion from the RDF-graph to the property graph is done via SciGraph, which loads OWL-Ontology-files into the Neo4j-database. Hereby, subjects and objects of the RDF-triples are converted into nodes, and predicates into edges. SciGraph also supports the implicit conversion from Compact URIs (CURIEs) to full IRIs for simpler queries.

The Planner Fast Downward is used to calculate the required operations of the robot. The data for the planner is queried from the Ontology-Server via Cypher and transferred to the planner via PDDL. The PDDL-files are generated using a template engine to integrate PDDL with the queried ontology data. The calculated abstract steps from the planner are parsed and enriched with additional information from the ontology to send concrete instructions to the robot via CRCL. This additional information includes the concrete coordinates of specific locations, information about the used tool, tolerances, etc.

Figure 12. Automated assembly of a PCB board.
5. Discussion of results

Cyber-Physical Systems are seen as a key technology for the development of future industrial systems. This paper introduces and demonstrates a model-driven approach combined with the concept of ontologies and explicit semantics for automatically configuring the control layer of CPS when applied in the assembly domain. More concrete, we developed and applied a similar knowledge intensive control architecture to tackle modeling, reasoning, planning and configuration aspects. The developed core architecture is applied for different purposes in the assembly domain such as automated configuration of an assembly line (pallet transport system) as well as for the control of an industrial robot.

The configuration approach relies on a system ontology, which contains sufficient information for automatically generating an FB network in the hardware-near LLC. In the presented work, only control software based on FBs can be represented in the ontology but it can be of any programming language or standard that supports such a concept. For increasing the generality and to encompass other control software programming paradigms (e.g. ladder logic or textual languages), the ontology needs to be extended with corresponding concepts. The presented approach is well-suited with regard to a modification of the assembly line’s physical layout. This requires only adapting the topology of the system stored in the resource ontology, which is a process needing a few minutes without elaborate programming efforts. Actually, the pallet transport system was modified once as a result of reorganizing the laboratory. The physical adaption required several hours of work but the control software adaption was performed in only a couple of minutes by updating the resource ontology. Performance measurements have shown that the automated configuration process is performed within approximately 1 second for the presented LLC applications. The configuration time is sufficiently fast enough for common industrial applications and remains roughly constant even in the case of several LLC applications being configured in parallel by the CP components. However, a component-oriented design of the LLC is a prerequisite to keep the configuration time within acceptable boundaries as has been shown in the comparison with the configuration time of a monolithic LLC design.

Moreover, the knowledge-driven approach enables that a robot can autonomously plan its actions and that functions such as automatic task selection are possible, thereby lowering the efforts significantly in the case of changes. A robot world model contains knowledge about all aspects of the robotics systems as well as products that needs to be assembled. The decision-making mechanism links the world model of the production environment and the robot skills. Based on automated reasoning, the robot can select the appropriate tool for a skill and perform a specific operation. The proposed solution can automate the robot configuration as well as the generation of process plans and schedules.

Generally, the proposed knowledge-driven architecture simplifies the engineering efforts. A system engineer only needs to describe the functionality of the assembly line or characteristics of the product to be assembled, without having to consider further engineering issues. The code for the industrial implementation is generated automatically based on the defined rules, states and actions. Through fast configuration and control code generation, the time for the design and development of automation systems can be reduced by a large extent and therefore a reduction of the engineering
costs is achieved. Besides, further functionalities such as reconfiguration can be added and managed with this architecture, which in this case need additionally triggering mechanism. In order to apply this architecture in other automation domains additional related modeling efforts are expected.

6. Conclusion

The assembly of customized products requires material flow systems as well as assembly equipment which are flexible and agile enough to accommodate dynamic and real-time changes in production processes. Auto-configuration as well as automated planning and execution capabilities of CPS in the manufacturing domain are regarded as of vital importance to achieve these requirements. Nevertheless, in current industrial practice, commonly the control system design is performed decoupled from hardware and software implementation details and the control applications are then developed manually for a designated system structure. Also typical industrial robot systems are not flexible enough to respond to the rapidly changing demands of new production processes and their growing complexity.

The model-driven engineering approach presented in this paper combines an ontology-based knowledge representation with CPS technology and uses a generative programming approach to automatically configure the hardware-near control applications of a distributed CPS in an assembly line. Thereby, the components of the CPS and their relations are specified in an ontology which acts as a part of the knowledge base of each component. Based on this representation, each component’s HLC utilizes configuration knowledge for performing a specification check and for carrying out the actual configuration by creating and wiring the software components (i.e. the IEC 61,499 FBs) of the solution space. Thereby, an executable LLC implementation is generated for each CPS component within the corresponding controller. The configuration time is sufficiently fast enough for common industrial applications and the approach supports the process of modifying the assembly line layout.

Furthermore, regarding the industrial robot use case, an ontology-based product model is used to link the used concepts linking product model, manufacturing process, and production infrastructure in a way that enables automatic reasoning and accurate and efficient robot operation. Similar CP component control architecture, as in previous use case, enables an automated robotic system to manufacture different types of products when autonomously perform reasoning (e.g. to select a specific tool) in order to perform a particular activity. The solution can automate the configuration as well as the generation of process plans and their execution, thereby lowering the efforts significantly in the case of changes.

As the presented work is centered on a knowledge-based approach, it can be adopted easily for different target systems by changing the contents of the ontology, respectively the problem space. Such modifications may encompass not only the physical components of the CPS but also the requirements and restrictions of the software components concerning their wiring to achieve the executable hardware-near control application. Future work will focus on the integration of these two use cases and ontology models. Also an extension of the existing ontology concept is envisaged as well as the integration of further assembly equipment in order to be able to test more complex scenarios.
Acknowledgments

The authors acknowledge the financial support from the “Production of the Future” program of the Austrian Ministry for Transport, Innovation and Technology under contract FFG 858707. Part of the research in this paper was performed during the authors’ previous engagement at the Automation and Control Institute (ACIN), Vienna University of Technology.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the the Austrian Ministry for Transport, Innovation and Technology under contract FFG [858707].

ORCID

Wilfried Lepuschitz http://orcid.org/0000-0003-2586-9813

References

Alsaﬁ, Y., & Vyatkin, V. (2010). Ontology-based reconﬁguration agent for intelligent mechatronic systems in ﬂexible manufacturing. Robotics and Computer-Integrated Manufacturing, 26 (4), 381–391.

Argyrou, A., Giannoulis, C., Papakostas, N., & Chryssolouris, G. (2016a). A uniform data model for representing symbiotic assembly stations. Procedia CIRP, 44, 85–90. 6th CIRP Conference on Assembly Technologies and Systems (CATS).

Argyrou, A., Giannoulis, C., Papakostas, N., & Chryssolouris, G. (2016b). A uniform data model for representing symbiotic assembly stations. Procedia CIRP, 44, 85–90. 6th CIRP Conference on Assembly Technologies and Systems (CATS).

Baheti, R., & Gill, H. (2011). Cyber-physical systems. In T. Samad & A. Annaswamy (Eds.), The impact of control technology (pp. pp. 161–166). IEEE Control Systems Society.

Balaji, B., Al Faruque, M. A., Dutt, N., Gupta, R., & Agarwal, Y. (2015, June). Models, abstractions, and architectures: The missing links in cyber-physical systems. In 2015 52nd acm/edac/ieee design automation conference (dac) (p. 1–6).

Bergert, M., Diedrich, C., Kiefer, J., & Bar, T. (2007). Automated PLC software generation based on standardized digital process information. In IEEE conference on emerging technologies and factory automation (pp. 352–359). Patras, Greece.

Bhrer, U. T., Legat, C., & Vogel-Heuser, B. (2015). Changeability of manufacturing automation systems using an orchestration engine for programmable logic controllers. IFAC-PapersOnLine, 48(3), 1573–1579. Retrieved from: http://www.sciencedirect.com/science/article/pii/S2405896315005492

Bonf, M., Fantuzzi, C., & Secchi, C. (2013) Retrieved from http://www.sciencedirect.com/science/article/pii/S0967066112000810 (Advanced Software Engineering in Industrial Automation (INCOM09). Design patterns for model-based automation software design and implementation. Control Engineering Practice, 21(11), 1608–1619.

Brennan, R. (2007). Toward real-time distributed intelligent control: A survey of research themes and applications. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 37(5), 744–765.
Brennan, R., Fletcher, M., & Norrie, D. (2002). An agent-based approach to reconfiguration of real-time distributed control systems. *IEEE Transactions on Robotics and Automation, 18*(4), 444–451.

Christensen, J. (2003). HMS/FB architecture and its implementation. In S. M. Deen (Ed.), *Agent-based manufacturing: Advances in the holonic approach* (pp. 53–88). Berlin, Germany: Springer-Verlag.

Ciccozzi, F., & Spalazzese, R. (2016, October). Mde4iot: Supporting the internet of things with model-driven engineering. In *10th international symposium on intelligent distributed computing*. Paris, France. Retrieved from [http://www.es.mdh.se/publications/4386-](http://www.es.mdh.se/publications/4386-)

Colla, M., Leidi, T., & Semo, M. (2009). Design and implementation of industrial automation control systems: A survey. In *IEEE international conference on industrial informatics* (pp. 570–575). Cardiff, Wales, UK.

Crosby, M., Petrick, R. P. A., & Toscano, C. (2016). Integrating mission, logistics, and task planning for skills-based robot control in industrial kitting applications. In *Proceedings of the 34th workshop of the uk planning and scheduling special interest group*. Huddersfield, UK.

Cucinotta, T., Mancina, A., Anastasi, G., Lipari, G., Mangeruca, L., Checozzo, R., & Rusinà, F. (2009). A real-time service-oriented architecture for industrial automation. *IEEE Transactions on Industrial Informatics, 5*(3), 267–277.

Czarnecki, K., & Eisenecker, U. (2000). *Generative programming: methods, tools, and applications*. Boston, MA, USA: Addison-Wesley.

Dai, W., Dubinin, V. N., Christensen, J. H., Vyatkin, V., & Guan, X. (2017, April). Toward self-manageable and adaptive industrial cyber-physical systems with knowledge-driven autonomic service management. *IEEE Transactions on Industrial Informatics, 13*(2), 725–736.

ElMaraghy, H., & ElMaraghy, W. (2016). Smart adaptable assembly systems. *Procedia CIRP, 44*, 4–13. 6th CIRP Conference on Assembly Technologies and Systems (CATS).

Facchinetti, T., & Della Vedova, M. (2011). Real-time modeling for direct load control in cyber-physical power systems. *IEEE Transactions on Industrial Informatics, 7*(4), 689–698.

Ferrer, B. R., Ahmad, B., Lobov, A., Vera, D., Lastra, J. L. M., & Harrison, R. (2015, July). A knowledge-based solution for automatic mapping in component based automation systems. In *2015 ieee 13th international conference on industrial informatics (indin)* (pp. 262–268). Cambridge, UK.

Fischer, J., Vogel-Heuser, B., & Friedrich, D. (2015, September). Configuration of plc software for automated warehouses based on reusable components- an industrial case study. In *2015 ieee 20th conference on emerging technologies factory automation (etaf)* (p. 1–7). Luxembourg.

Garca, C. A., Castellanos, E. X., & Garca, M. V. (2018). Uml-based cyber-physical production systems on low-cost devices under iec-61499. *Machines, 6*(2). Retrieved from [http://www.mdpi.com/2075-1702/6/2/22](http://www.mdpi.com/2075-1702/6/2/22)

González, E., Hamilton, A., Moreno, L., Marichal, R., Munoz, V., & Seliger, G. (2006). Software experience when using ontologies in a multi-agent system for automated planning and scheduling. *Software: Practice and Experience, 36*(7), 667–688.

Gritzner, D., & Greenyer, J. (2018). Generating correct, compact, and efficient PLC Code from scenario-based assume-guarantee specifications. *Procedia Manufacturing, 24*, 153–158. Retrieved from [http://www.sciencedirect.com/science/article/pii/S2351978918305651](http://www.sciencedirect.com/science/article/pii/S2351978918305651)

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*(2), 199–220.

Han, K. H., & Park, J. W. (2009). Object-oriented ladder logic development framework based on the unified modeling language. In J. Kacprzyk, R. Lee, G. Hu, & H. Miao (Eds.), *Computer and information science 2009* (Vol. 208, pp. 33–45). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hehenberger, P., Vogel-Heuser, B., Bradley, D., Eynard, B., Tomiyama, T., & Achiche, S. (2016). Design, modelling, simulation and integration of cyber physical systems: Methods and applications. *Computers in Industry, 82*, 273–289. Retrieved from [http://www.sciencedirect.com/science/article/pii/S0166361516300902](http://www.sciencedirect.com/science/article/pii/S0166361516300902)
Huckaby, J., Vassos, S., & Christensen, H. I. (2013). Planning with a task modeling framework in manufacturing robotics. In 2013 ieee/rsj international conference on intelligent robots and systems (p. 5787–5794). Tokyo, Japan.

Iarovyi, S., Ramis, B., Xiangbin, X., Sampath, A., Lobov, A., & Lastra, J. L. M. (2015). Representation of manufacturing equipment and services for okd-mes: From service descriptions to ontology. In 2015 ieee 13th international conference on industrial informatics (indin) (p. 1069–1074). Cambridge, UK.

IEC. (1993). Iec 61131-3 programmable controllers – Part 3: Programming languages. Geneva: International Electrotechnical Commission.

IEC. (1997). IEC 61512-1 batch control – part 1: Models and Terminology. Geneva: International Electrotechnical Commission.

IEC. (2005). Iec 61499-1 function blocks – part 1: Architecture. Geneva: International Electrotechnical Commission.

Jade - Java Agent Development Framework. (2018). Retrieved from http://jade.tilab.com (accessed December 2018).

Jamro, M. (2014, September). Automatic generation of implementation in sysml-based model-driven development for iec 61131–3 control software. In 2014 19th international conference on methods and models in automation and robotics (mmar) (p. 468–473). Miedzyzdroje, Poland.

Julius, R., Schrenberg, M., Schumacher, F., & Fay, A. (2017). Transformation of GRAFCET to PLC code including hierarchical structures. Control Engineering Practice, 64, 173–194. Retrieved from http://www.sciencedirect.com/science/article/pii/S0967066117300709

Kang, W., Kapitanova, K., & Son, S. (2012). RDGS: A real-time data distribution service for cyber-physical systems. IEEE Transactions on Industrial Informatics, 8(2), 393–405.

Kim, K.-D., & Kumar, P. (2012). Cyber-physical systems: A perspective at the centennial. Proceedings of the IEEE, 100(Special Centennial Issue), 1287–1308.

Kim, K.-Y., Manley, D. G., & Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. Computer-Aided Design, 38(12), 1233–1250.

Kootbally, Z., Kramer, T. R., Schlenoff, C., & Gupta, S. K. (2017). Overview of an ontology-based approach for kit building applications. In 2017 ieee 11th international conference on semantic computing (icsc) (p. 520–525). San Diego, CA.

Kramer, J., & Magee, J. (2007). Self-managed systems: An architectural challenge. In Future of software engineering (pp. pp. 259–268). Washington, DC: IEEE Computer Society.

Krzysztof, S. (2005). Automatic code generation for PLC controllers. In R. Winther, B. Gran, & G. Dahl (Eds.), Computer safety, reliability, and security (Vol. 3688, pp. pp. 303–316). Berlin, Heidelberg: Springer Berlin Heidelberg.

Lanthaler, M. (n.d.). Self-healing wireless sensor networks. Retrieved from https://www.cs.helsinki.fi/u/niklande/opetus/SemK07/review/lanthaler.pdf

Lastra, J., & Delamer, M. (2006). Semantic web services in factory automation: fundamental insights and research roadmap. IEEE Transactions on Industrial Informatics, 2(1), 1–11.

Lee, E. (2008). Cyber physical systems: Design challenges. In 11th IEEE international symposium on object oriented real-time distributed computing (pp. 363–369). Orlando, FL.

Lee, S., Ang, M. A., & Lee, J. (2006). Automatic generation of logic control (Tech. Rep.). Loughborough University/University of Michigan.

Legat, C., Lamparter, S., & Vogel-Heuser, B. (2013). Knowledge-based technologies for future factory engineering and control. In T. Borangiu, A. Thomas, & D. Trentesaux (Eds.), Service orientation in holonic and multi agent manufacturing and robotics (pp. pp. 355–374). Berlin, Heidelberg: Springer Berlin Heidelberg.

Legat, C., Schtz, D., & Vogel-Heuser, B. (2014, October). Automatic generation of field control strategies for supporting (re)-engineering of manufacturing systems. Journal of Intelligent Manufacturing, 25(5), 1101–1111. Retrieved from https://link.springer.com/article/10.1007/s10845-013-0744-z

Leitao, P. (2004). An agile and adaptive holonic architecture for manufacturing control (PhD thesis). University of Porto, Portugal.
Leitao, P. (2009, October). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence, 22*(7), 979–991.

Lemaignan, S., Siadat, A., Dantan, J.-Y., & Semenenko, A. (2006). MASON: A proposal for an ontology of manufacturing domain. In *Ieee workshop on distributed intelligent systems: Collective intelligence and its applications* (pp. 195–200). Prague, Czech Republic.

Lepuschitz, W., Vallée, M., Zoitl, A., & Merdan, M. (2010). Online reconfiguration of the low level control for automation agents. In *36th annual conference on ieee industrial s society* (pp. 1365–1370). Glendale, AZ.

Lepuschitz, W., Zoitl, A., & Merdan, M. (2011). Ontology-driven automated software configuration for manufacturing system components. In *IEEE international conference on systems, man, and cybernetics, Part C: Applications and Reviews, 41*(1), 52–69.

List, E. (2015). *Eine Assembling-Ontologie für wissensbasierte Systeme* (master’s thesis). Vienna University of Technology, Austria.

Lohse, N., Hirani, H., & Ratchev, S. (2006). Equipment ontology for modular reconfigurable assembly systems. *International Journal of Flexible Manufacturing Systems, 17*(4), 301–314.

López Orozco, O. J., & Lastra, J. L. M. (2006). Using semantic web technologies to describe automation objects. *International Journal of Manufacturing Research, 1*(4), 482–503.

Mcdermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., & Wilkins, D. (1998). *PDDL - the planning domain definition language* (Tech. Rep.). CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Merdan, M. (2009). Knowledge-based multi-agent architecture applied in the assembly domain (PhD thesis). Vienna University of Technology, Austria.

Merdan, M., Vallée, M., Lepuschitz, W., & Zoitl, A. (2011). Monitoring and diagnostics of industrial systems using automation agents. *International Journal of Production Research, 49*(5), 1497–1509.

Metzger, M., & Polaków, G. (2011). A survey on applications of agent technology in industrial process control. *IEEE Transactions on Industrial Informatics, 7*(4), 570–581.

Michalos, G., Makris, S., & Chryssolouris, G. (2015). The new assembly system paradigm. *International Journal of Computer Integrated Manufacturing, 28*(12), 1252–1261.

Morán, G., Pérez, F., Orive, D., Estévez, E., & Marcos, M. (2011). Automatic composition of IEC 61499 distributed control applications. In *Ieee international conference on emerging technologies and factory automation* (pp. 1–8). Toulouse, France.

Negri, E., Fumagalli, L., Garetti, M., & Tanca, L. (2016). Requirements and languages for the semantic representation of manufacturing systems. *Computers in Industry, 81*, 55–66. Retrieved from [http://www.sciencedirect.com/science/article/pii/S0166361515300555](http://www.sciencedirect.com/science/article/pii/S0166361515300555)

Patel, R., Hedelind, M., & Lozan-Villegas, P. (2012). Enabling robots in small-part assembly lines: The “rosetta approach” - an industrial perspective. In *Robotik 2012; 7th german conference on robotics* (p. 1–5). Munich, Germany.

Pavlovskyi, Y., Kennel, M., & Schmucker, U. (2018, November). Template-based generation of plc software from plant models using graph representation. In *2018 25th international conference on mechatronics and machine vision in practice (m2vip)* (p. 1–8). Stuttgart, Germany.

Pedersen, M. (2014). *Robot skills for transformable manufacturing systems* (PhD thesis). Aalborg University, Denmark.

Perzylo, A., Somani, N., Profanter, S., Kessler, I., Rickert, M., & Knoll, A. (2016). Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. In *2016 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 2293–2300). Daejeon, South Korea.

Perzylo, A., Somani, N., Rickert, M., & Knoll, A. (2015). An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions. In *2015 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 4197–4203). Hamburg, Germany.
Priego, R., Schtz, D., Vogel-Heuser, B., & Marcos, M. (2015, September). Reconfiguration architecture for updates of automation systems during operation. In 2015 IEEE 20th conference on emerging technologies factory automation (etfa) (p. 1–8). Luxembourg.

Proctor, F., Balakirsky, S., Kootbally, Z., Kramer, T., Schlenoff, C., & Shackleford, W. P. (2016, August). The canonical robot command language (crcl). Industrial Robot: An International Journal, 43, 495–502.

Rabemanantsoa, M. (1993). Knowledge-based system for assembly process-planning. In Proceedings 1993 software engineering standards symposium (p. 267–272). Brighton, UK.

Ramis Ferrer, B., Jarovyi, S., Lobov, A., Martinez Lastra, J., & Mohammed, W. (2016). Exemplifying the potentials of web standards for automation control in manufacturing systems. International Journal of Simulation Systems, Science & Technology, 17(33), 3.1–3.12.

Regulin, D., Schütz, D., Aicher, T., & Vogel-Heuser, B. (2016). Model based design of knowledge bases in multi agent systems for enabling automatic reconfiguration capabilities of material flow modules. 2016 IEEE International Conference on Automation Science and Engineering (CASE), 133–140. Fort Worth, TX, USA.

Reinhart, G., Krug, S., Hüttner, S., Mari, Z., Riedelbauch, F., & Schlögel, M. (2010). Automatic configuration (Plug & Produce) of Industrial Ethernet Networks. In 9th IEEE international conference on industry applications (pp. 1–6). Sao Paulo, Brazil.

Ringert, J. O., Roth, A., Rumpe, B., & Wortmann, A. (2015). Code generator composition for model-driven engineering of robotics component & connector systems. CoRR, abs/1505.00904 . Retrieved from http://arxiv.org/abs/1505.00904

Saber, A., & Kumar, G. (2010). Efficient utilization of renewable energy sources by gridable vehicles in cyber-physical energy systems. IEEE Systems Journal, 4(3), 285–294.

Steinegger, M., & Zoitl, A. (2012). Automated code generation for programmable logic controllers based on knowledge acquisition from engineering artifacts: concept and case study. In IEEE international conference on emerging technologies and factory automation (pp. 1–8). Krakow, Poland.

Stenmark, M. (2011). Integration of semantic knowledge to enable re-use of robot programs (master’s thesis). Lund University, Sweden.

Stenmark, M., Haage, M., & Topp, E. A. (2017). Simplified programming of re-usable skills on a safe industrial robot – Prototype and evaluation. In Proceedings of the ieee/acm conference on human-robot interaction (hri) (pp. 463–472). Vienna, Austria: ACM. doi:10.1145/2909824.3020227

Stenmark, M., & Malec, J. (2015). Knowledge-based instruction of manipulation tasks for industrial robotics. Robotics and Computer-Integrated Manufacturing, 33, 56–67. Special Issue on Knowledge Driven Robotics and Manufacturing.

Thramboulidis, K., Bochalis, P., & Bouloumpasis, J. (2017). A framework for mde of iot-based manufacturing cyber-physical systems. In Proceedings of the seventh international conference on the internet of things (pp. 11:1–11:8). New York, NY: ACM. doi:10.1145/3131542.3131554

Thramboulidis, K., & Christoulakis, F. (2016, October). UML4iotA UML-based approach to exploit IoT in cyber-physical manufacturing systems. Computers in Industry, 82, 259–272. Retrieved from http://www.sciencedirect.com/science/article/pii/S016636151630094X

Tikhonov, D., Schtz, D., Ulewicz, S., & Vogel-Heuser, B. (2014, October). Towards industrial application of model-driven platform-independent plc programming using uml. In Iecon 2014-40th annual conference of the ieee industrial electronics society (p. 2638–2644). Dallas, TX.

Vallée, M., Kaindl, H., Merdan, M., Lepuschitz, W., Arnautovic, E., & Vrba, P. (2009). An automation agent architecture with a reflective world model in manufacturing systems. In IEEE international conference on systems, man, and cybernetics (pp. 305–310). San Antonio, Texas, USA.
Vallée, M., Merdan, M., Lepuschitz, W., & Koppensteiner, G. (2011). Decentralized reconfiguration of a flexible transportation system. *IEEE Transactions on Industrial Informatics, 7*(3), 505–516.

Vicaire, P., Hoque, E., Xie, Z., & Stankovic, J. (2012). Bundle: A group-based programming abstraction for cyber-physical systems. *IEEE Transactions on Industrial Informatics, 8*(2), 379–392.

Vogel-Heuser, B., Fischer, J., Feldmann, S., Ulewicz, S., & Rsch, S. (2017). Modularity and architecture of plc-based software for automated production systems: An analysis in industrial companies. *Journal of Systems and Software, 131*, 35–62. Retrieved from [http://www.sciencedirect.com/science/article/pii/S0164121217300985](http://www.sciencedirect.com/science/article/pii/S0164121217300985)

Vogel-Heuser, B., Schtz, D., Frank, T., & Legat, C. (2014). Model-driven engineering of manufacturing automation software projects: A SysML-based approach. *Mechatronics, 24*(7), 883–897. Retrieved from: [http://www.sciencedirect.com/science/article/pii/S0957415814000853](http://www.sciencedirect.com/science/article/pii/S0957415814000853)

Vogel-Heuser, B., Witsch, D., & Katzke, U. (2005). Automatic code generation from a UML model to IEC 61131-3 and system configuration tools. In *IEEE international conference on control and automation* (Vol. 2, pp. 1034–1039). Budapest, Hungary.

Vyatkin, V. (2011). IEC 61499 as enabler of distributed and intelligent automation SOTA review. *IEEE Transactions on Industrial Informatics, 7*(4), 768–781.

Vyatkin, V. (2013, August). Software engineering in industrial automation: State-of-the-art review. *IEEE Transactions on Industrial Informatics, 9*(3), 1234–1249.

Wang, L.-C., & Lin, S.-K. (2009). A multi-agent based agile manufacturing planning and control system. *Computers & Industrial Engineering, 57*(2), 620–640.

Wolf, W. (2009). Cyber-physical Systems. *Computer, 42*(3), 88–89.

Zander, S., Ahmed, N., & Hua, Y. (2016). Empowering the model-driven engineering of robotic applications using ontological semantics and reasoning. 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Porto, Portugal. In *Keod*.

Zoitl, A. (2009). *Real-Time Execution for IEC 61499*. USA: ISA-o3neidaA.

Zoitl, A., Lepuschitz, W., Merdan, M., & Vallée, M. (2010). A real-time reconfiguration infrastructure for distributed embedded control systems. In *Ieee international conference on emerging technologies and factory automation* (pp. 1–8). Bilbao, Spain.