Deploying Microservices for a Cloud-based Design of System-of-Systems in Intralogistics

Andreas Habl, Orthodoxos Kipouridis, Johannes Fottner
Institute for Materials Handling, Material Flow, Logistics, Technical University of Munich
Munich, Germany
{habl, kipouridis, fottner}@fml.mw.tum.de

Abstract — **For the realization of flexible, reconfigurable manufacturing and logistics facilities, heterogeneous, autonomous, and interconnected intralogistics systems are being deployed on the shop-floor. However, the distributed and often emerging nature of such systems makes their design and integration into existing processes a rather complex engineering task that requires a lot of coordination from all stakeholders. Modern IT tools offer solutions, but often target only individual system aspects such as modeling and simulation and rarely address the continuous evolution and changing requirements of such intralogistics systems. In this work, we approach distributed intralogistics systems as a system of systems and present a cloud-based engineering design process using microservices. In order to enable distributed simulation and testing of intralogistics systems, we employ a service-oriented architecture at systems' model-level, to couple simulation engines through a cloud-based platform. The applicability of the concept is demonstrated by a case study using a Discrete Event System Specification simulation framework, presenting system configuration and interface specification at run-time.**

Keywords — System of Systems, Microservices, Intralogistics, Cloud-based Design, DEVS

I. Introduction

Modern consumer trends such as the increasing demand for highly customized products and shorter product cycles pose significant challenges to manufacturing and intralogistics systems, where intralogistics describes the organization, realization and optimization of internal material flow and logistic technologies [1]. In order to cope with requirements for high flexibility and adaptability, independent, decentralized controlled systems are being deployed in modern production and logistics facilities. Such Cyber-physical Systems (CPS) are characterized by a close coupling of computing and physical elements that enables them to sense their environment and independently pursue their goals [2]. Multiple CPS can be merged into larger systems, forming a System of Systems (SoS), that demonstrate new characteristics like emerging behavior and evolutionary development [3].

Today, CPS-based manufacturing and intralogistics systems are being deployed towards realization of flexible logistics facilities. However, the adoption of such SoS seems to be hitting barriers regarding achieving interoperability and conformity between the constituent systems. The intrinsic properties of SoS such as heterogeneity, independence and geographical distribution of the individual systems in combination with emergent behaviors that allow SoS to evolve and adapt to dynamic environments make the processes of designing a SoS to achieve interoperability a rather challenging one. This situation is additionally enhanced due to the lack of globally accepted standards as well as the diversity of interfaces that are used in this area. Achieving interoperability is particularly challenging in the case of intralogistics facilities, where management and ownership of systems is distributed among multiple parties and the integration with various legacy systems is a prerequisite for successful deployment. With a view to address the above challenges, Service-Oriented Architectures (SOA) have been introduced to the control level of industrial systems [4]. SOA-based approaches enable the integration of heterogeneous applications through services and can be applied to industrial systems to support cross-layer interoperability. Nonetheless, enabling distributed, heterogeneous, SOA-based systems to interoperate is still a complex engineering task involving configuration at multiple levels (e.g. networking, operational, layout). The above aspects highlight the need for collaboration-oriented concepts and tools that support collaboration among stakeholders and allow the planning and analysis of the behavior of a SoS not only prior to deployment but also during operation and at run-time.

In this work, we present a cloud-based approach for designing SoS in intralogistics aiming to support interoperability. We take advantage of the loose coupling principles of SOA to expose the functionalities and interfaces of constituent systems. Using service-based architectures we describe how cloud-based engineering design tools can support a collaborative process of designing, integrating and visualizing SoS. The concept is demonstrated by a case study connecting a robot with two Discrete Event System Specification (DEVS) [5] simulation engines running models of a conveyor, and configuring them so that they can execute logistics tasks.

This paper is organized as follows. In the next section, background information and related work in the field of SoS, SOA-based systems, and microservices in a cloud-based design is presented in the context of intralogistics systems. The third section describes the proposed concept of leveraging a distributed platform for enabling distributed integration, simulation, and testing. Following, in the fourth chapter a case study is presented, demonstrating the collaborative design of a

SoS. Finally, the paper concludes with a discussion on the potential, challenges, and future work of the presented approach.

II. Background and Related Work

A. *System of Systems*

As the importance of integrating CPS in the shop-floor increases, so is the need for developing efficient processes for their engineering and integration with existing systems towards a SoS. Consequently, a SoS is defined as heterogeneous, autonomous systems that are networked together in order to achieve a common goal [3].

When it comes to the engineering of SoS, interoperability between constituent systems refers to the capability of those systems to communicate, exchange information and matter, and collaborate for accomplishing a common task. Since individual systems of a SoS can evolve over time, focus should be placed on configuring interfaces between the systems to enable interoperability. In this sense, proper methodologies and tools are needed to ensure communication across heterogeneous interfaces. An effective way to achieve this is by using SOA, which has been already applied to SoS for instance in areas like energy management in [6].

B. *Modeling and Testing System Functionalities through Service-Oriented Architectures*

A factor that led to the success of deploying SOA-oriented systems and web services is the high level of standardization in this area, using standards such as Web Services Description Language (WSDL) which is an interface definition language for describing functionality of a web service. Using the concept of SOA with the technology of web services implies several benefits. Services are available over network and can be discovered through the service registry (UDDI). Additionally, the published interfaces allow for calling a service without knowledge of its implementation, thus providing interoperability between different platforms of service consumers and providers. The application of SOA within an industrial SoS is illustrated in figure 1, where the coupling of a CPS and a simulated system is achieved by web services. The concept of using services for exposing functionalities of individual systems has been introduced to implement decentralized controlled system architectures in the area of the Industrial Internet of Things, with the view of realizing modular, reconfigurable manufacturing facilities [7]. In the field of industrial automation, SOA-oriented approaches are applied in various areas such as logistic and production systems, and often implemented by knowledge-based web services [8]. In order to enable evolvable and collaborative production systems by addressing the required flexibility and reconfigurability, a service-oriented production control architecture has been introduced in [9].
Nevertheless, there is a lot of configuration and testing effort that has to be put in order to achieve interoperability. A major challenge is that configuration requires collaboration-intense tasks, such as the specification and agreeing on interfaces to integrate independently developed systems, often without having the authority for significant redesign of the total system, which presents a big challenge. Moreover, the application area of designing and implementing intralogistics systems often requires geographically distributed teams with different backgrounds, such as control engineers, logistic designers, and system integrators to effectively work together on a project basis.

In this direction, Modeling and Simulation (M&S) tools play a central role in supporting engineering tasks at various phases of a SoS design process. Concepts towards co-modeling and co-simulation of SoS have been presented that include agent- and event-based approaches [10, 11]. However, these approaches aim at providing single-user M&S tools for developing compatible system's models and simulation environments, and rarely support a dynamic, simultaneous configuration of running models in a distributed (simulation) environment.

Recognizing the need for a distributed simulation among multiple simulation engines, approaches of integrating DEVS with SOA (DEVS/SOA) have been introduced [12]. The DEVS/SOA approach describes a process where different clients upload their respective models into the remote simulation server, compile the models, and execute the simulation in a serialized manner. However, this simulation-as-a-service approach requires the existence and availability of a net-centric infrastructure with one or more (DEVS) simulation servers. For intralogistics joint projects, the provision of such an infrastructure can be rather costly and time-consuming. Therefore, appropriate ICT tools are required that enable a distributed design on a project basis.

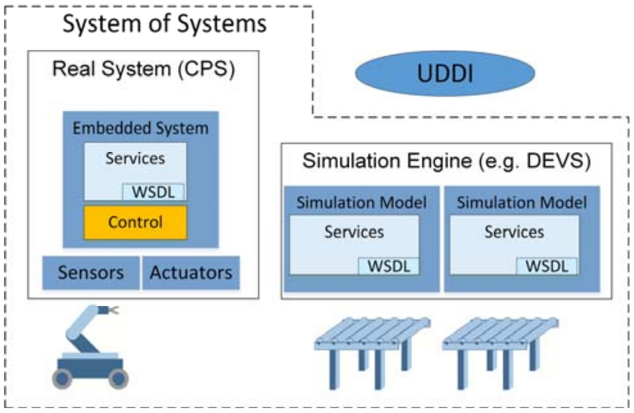

Fig. 1. SOA-based SoS composed of a real and a simulated system

C. *Cloud-based Design and Microservices*

The design of intralogistics facilities in joint projects is a multidisciplinary activity that can be divided in four phases, namely preparation, rough planning, detailed planning and realization [13]. Typically, in each phase different software tools are used to support the various engineering tasks. Today, digital, collaborative-oriented approaches are being used by engineering teams, particularly for the product design. However, these tools just focus on single aspects of the system development activities and do not specifically target the intralogistics application domain.

Originating from the field of product development, cloud-based design describes a service-oriented and networked design model that makes use of cloud computing and SOA technologies to support engineering and design services in distributed and collaborative environments [14]. For an ICT system to be characterized as cloud-based, it has to be based on cloud-computing models (SaaS, IaaS, PaaS), must be ubiquitously accessible and should be able to handle complex information flows.

In the area of distributed and cloud-based systems, microservices have been gaining a lot of attention in the past years. Lewis and Fowler describe microservices as an architectural style for developing an application as a suite of small services, each of which is running on its own process and communicating with lightweight mechanisms [15]. Although SOA as well as microservices architecture is considered as service-based architecture they present significant differences. Since each microservice is used to handle a single task, has its own data space, and can be deployed and scaled independently, this is particularly useful in the direction of developing customizable ICT tools and lightweight clients for supporting the design of SoS that can adapt to variable project requirements. But when it comes to integration of multiple heterogeneous systems and services, microservices architecture is at a disadvantage compared to SOA because of reduced number of choices for services integration where SOA has no upper limit [16].

III. Enabling a Cloud-based Design Using Microservices

In this section, we present a cloud-based design approach for the engineering of SoS in the application domain of intralogistics. The goal is to support the engineering of heterogeneous intralogistics systems prior to and after their deployment, by supporting a cross-company, collaborative design of SoS.

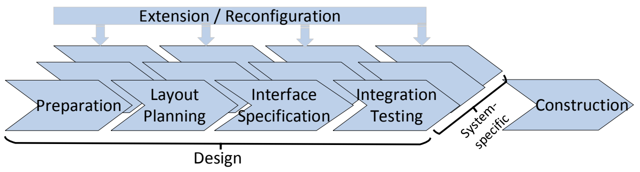

Fig. 2. Design process of a SoS-based intralogistics system

Modifying the design process of intralogistics systems as presented in [17], figure 2 presents a collaborative process that involves multiple stakeholders in the synchronous design of SoS-based intralogistics systems. This multi-step process is described below:

1. Preparation: The individual systems and system models are modeled project-independently from their respective manufacturers or suppliers. Their functions are published as services using WSDL. 3D models of the systems are to be used for the layout planning. Real systems and simulation engines are coupled with the cloud platform through cluster clients, so that they can be accessed by the network.

2. Layout planning: By using the shared resources of the previous step, 3D models, the physical environment and existing infrastructure is considered to define the facility's final layout. Collaborative web- or 3D virtual environments are deployed to promote collaboration between stakeholders such as logistic planners, manufacturers and customers.

3. Interface specification: To achieve interoperability, communication interfaces have to be specified and configured to enable the communication between the systems. At this phase, each system function is published by the individual service. A collaborative configuration of service interfaces would include the definition of the service name and the respective details like input/output parameters and a service description.

4. Integration testing: Set up a remote connection between systems for in-house testing. The functions of real systems can be remotely called from the users or other systems to drive simulation and test the interoperability of the overall system. Each function can be called asynchronously and at run time.

5. For supporting an ongoing expansion and reconfiguration of the overall intralogistics system, it should be possible that new subsystems can be added or be removed at any time.

To provide support for the above design process, our approach takes advantage of the loose coupling that SOA-enabled SoS offer, and combines it with the distributed nature of a microservice architecture, to implement a platform that facilitates the deployment of cloud-based, software-as-a-service tools.

A concept that implements a service-based architecture by using microservices and SOA is illustrated in figure 3. Each collaborative engineering tool aims to offer support to individual steps in the design process. The functionality of each tool is then implemented through individual microservices that can be in turn deployed separately at different time frames, from different providers and on a per-project basis. In order to implement this concept, sample tools were developed including browser-based facility layout planning, a 3D collaborative virtual environment for defining and agreeing upon the interface specifications and a distributed simulation.

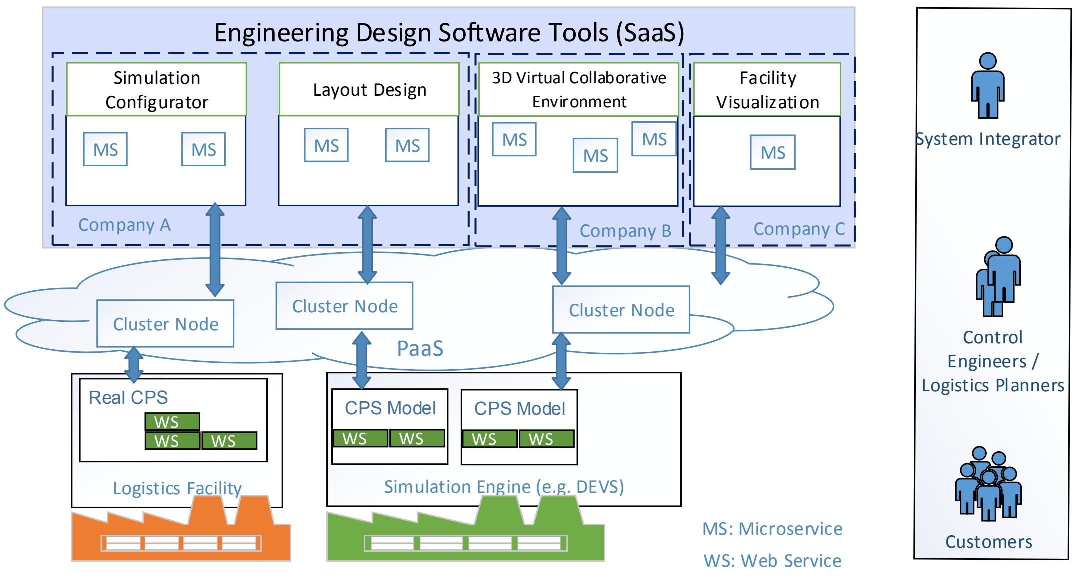

Fig. 3. Service-based architecture for designing SoS in the cloud

Each of these tools-as-service can be deployed from different parties and in different project phases. It should be however the case that the designated tools are available before the beginning of their respective phase. The tools can be deployed over private or public clouds and access is given to the project partners such as logistics planners, system integrators and facility operators, based on their role.

The data generated or modified from instances of one design tool have to be synchronized and made available to other services. For the data synchronization, we leverage an in-memory data grid technology (IMDG). In this, system specific data can be shared in a network creating a cluster of nodes, which provides a distributed data storage and a service registry, for handling concurrency among distributed data structures (hashmaps, queues, etc.). A prerequisite or the correct functionality is knowledge from the system supplier (or integrator) of the data model that is being used for modeling the system-specific data (status, timing etc.). A custom model was used in the prototypical implementation of the platform. The lack of unified standards in this area means that this information has to become available beforehand, and in a project-basis. It should however be mentioned that each SoS is unique from its nature. After distributing the system-related data in the cluster, each of the microservices can access a cluster node. The nodes execute queries on the dataset and can be agnostic of the overlying logic of each service implementation and can be started and stopped in isolation without impacting the cluster, either from a storage or from a messaging standpoint. This way applications can be split into small cooperating components that communicate with each other over standard protocols. The proposed concept is not dependent of the IMDG implementation. In this work, we use the solution of Hazelcast [18], other solutions like apache ignite or oracle coherence could also be considered. An early attempt to employ an IMDG for implementing a collaborative, open-source, software platform was presented in [19]. However, functionalities such as distributed simulation were yet to be studied or implemented.

In this paper we focus on enabling a distributed simulation using the above architecture. By leveraging the distributed characteristics of SOA-based CPS, integration and simulation capabilities can be provided to the deployed engineering tools. For CPS, services can run directly on embedded systems. SOA can provide the integration of low-level embedded system functions (sensors, actuators) and high-level information system services. Each CPS can then join the cluster and share their data in the network.

In order to test and analyze the behavior of SoS, we integrate this service-oriented functionality into the DEVS modeling and simulation framework, and publish the services through the simulation engine using standard web technologies. Executing services directly on each simulated system model is consistent with the goal to make an overall SoS function-oriented, and comes in contrast to a web-based simulation-as-a-service approach. The proposed architecture is not restrictive in terms of the simulation engine and/or modeling language. Thus, many of the model-based engineering approaches and tools can be used. With the support of the distributed platform, M&S tools can be used for simulating SOA system behavior, and configured to accommodate interoperability between systems modeled and simulated across multiple instances. This configuration refers to many functional aspects for intralogistics systems. For instance, configuring a handover process between two systems would include specification of timing, availability and routing information for each system.

IV. CASE STUDY: DESIGNING AND INTEGRATING A SOS IN INTRALOGISTICS

To test the feasibility of the above design process, a case study is presented in this section where we apply the process steps in a hypothetical use-case scenario in intralogistics. In this, two simulated conveyor systems are integrated and configured through the platform to work together. Goal of this study is to test functionality of the deployed tools and examine the integration of multiple simulation engines (here: DEVS), thus, proving the feasibility of a distributed simulation. Furthermore, we want to address the challenge of continuous evolution of SoS in intralogistics by allowing for deployment of the engineering tools in various system design phases as well as a system's reconfiguration at (simulation) run time.

We consider a setup in a logistics facility that consists of a robot that is distributing goods and a series of autonomous CPS-based flexible conveyor systems, namely FlexConveyors [20]. A FlexConveyor consists of single conveyor units that are connected to each other to form a complex conveyor system. The system as a whole can be considered as a SoS and due to its heterogeneity, modular design and decentralized control it is a representative system of a SoS in intralogistics.

Based on the multistep design process as presented in the previous section, the design starts with the preparation phase. In this, the 3D models for every individual system are generated as well as we have ensured that these systems are able to expose their functionalities in services by using WSDL. To complete the first step the individual systems as well as simulation models are coupled with the platform. For that we assign a cluster client to each system. In our case, there is a picking robot, and multiple simulated system models of the conveyor systems. As illustrated in fig. 4a, conveyor systems are split up into two DEVS simulation engines and so we obtain one DEVS simulation engine that is handling incoming containers and another simulation engine that is handling outgoing containers after they are sorted by the robot. We use the java-based DEVS-Suite [21] to model and simulate the conveyor systems. However, other M&S engines and tools could also be considered as long as they support or can be modified to support SOA-based communication.

In a next step, planning the layout of the overall intralogistics system is done by placing and arranging the available 3D models of the conveyor systems and the robot. For this, a web-based tool was developed and coupled to the platform. As it is illustrated in figure 4b, the robot is located between the conveyor rows to perform its distribution tasks. Moving on in the design process, it follows the interface specification. Here, the connections between individual systems are specified that define the relationships between the simulation models and/or systems. As shown in figure 4c, the connection between the conveyor systems and the robot can be defined in the 3D interactive environment tool. This step includes configuration of the single service interface of each component system. In particular, the system functions like an availability check or handover operation of the conveyor units are published. In order to guarantee end-to-end transport of goods, a routing service was implemented and published as a service, which can be called by the individual conveyor units.

The last step involves testing the integration of the designed SoS. To achieve that, a distributed simulation can be executed where multiple simulation engines communicate over the network. Each individual system (model or real) should be in the position to call the right functions to perform a handover task, for example between conveyor system and robot. The goal is to execute end-to-end transport orders and to test their error-free execution (see figure 4d). In a running session, the topology of the overall system can be modified, and real or simulated systems can be added or removed. Thus it is possible to test effects of reconfiguration on system behavior at runtime.

The above case study demonstrated the cloud-based design of a heterogeneous intralogistics system using individually deployed tools, to achieve interoperability of the various systems. Our approach achieved the simulation and testing of different systems in a distributed manner. Results showed successful handovers and end-to-end conveying of packages using loosely coupled services.

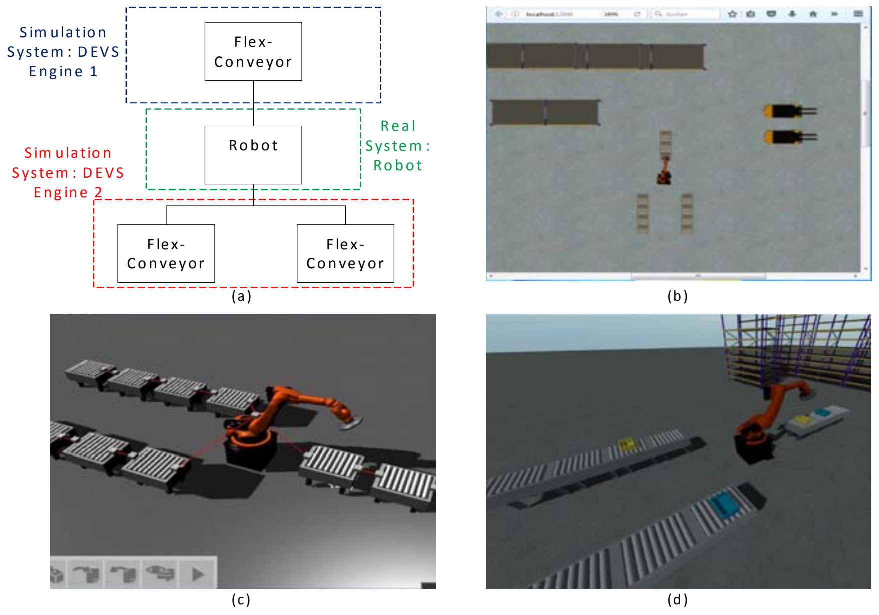

Fig. 4. Setup of the case study: (a) schematic, (b) web browser-based layout editor, (c) defining connections between systems in a java application, (d) multi-user 3D visualization of a running simulation in Unity3D

V. Conclusion

In this paper, we presented a cloud-based approach for designing heterogeneous intralogistics systems that leverages microservices as well as SOA. This approach enables the deployment of distributed collaboration software tools, such as multi-user layout planning, visualization and distributed simulation on a project basis. As a side effect when using microservices, the level of complexity of the edge components and infrastructure (i.e. single software tools and automation systems) is increased. This requires that the engineering teams have a good understanding of the various system components that cooperate as well as the ways they communicate with each other.

The presented approach aims to support the design process of SoS in intralogistics facilities. For future work, further research that targets the field of collaboration engineering and social aspects of the design tasks is required. Other areas where heterogeneous CPS find application, such as manufacturing or maintenance systems can also be considered.

References

[1] Logistics Journal, *Intralogistics.* [Online] Available: https://www.logistics-journal.de/about/intralogistics. Accessed on: Jun. 06 2017.

[2] Acatech, *Cyber-Physical Systems: Driving force for innovations in mobility, health, energy and production*. Dordrecht: Springer, 2012.

[3] M. Jamshidi, Ed., *System of systems engineering: Innovations for the 21st century*. Hoboken, N.J: Wiley, 2009.

[4] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bangemann, "Towards an architecture for service-oriented process monitoring and control," in *IECON 2010:* IEEE, 2010, pp. 1385–1391.

[5] A. I. Concepcion and B. P. Zeigler, "DEVS formalism: A framework for hierarchical model development," *IIEEE Trans. Software Eng.*, vol. 14, no. 2, pp. 228–241, 1988.

[6] D. Mora, M. Taisch, A. W. Colombo, and J. M. Mendes, "Service-oriented architecture approach for industrial System of Systems: State-of-the-Art for Energy Management," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 1246–1251.

[7] A.-W. Colombo, S. Karnouskos, and J.-M. Mendes, "Factory of the Future: A Service-oriented System of Modular, Dynamic Reconfigurable and Collaborative Systems," in *Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management*: Springer London, 2010, pp. 459–481.

[8] B. Ramis *et al.,* "Knowledge-based web service integration for industrial automation," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 733–739.

[9] J. M. Mendes, P. Leitao, A. W. Colombo, and F. Restivo, "Service-oriented control architecture for reconfigurable production systems," in *2008 6th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 744–749.

[10] J. M. Mendes, P. Leitão, F. Restivo, and A. W. Colombo, "Service-Oriented Agents for Collaborative Industrial Automation and Production Systems," in *Holonic and Multi-Agent Systems for Manufacturing: 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2009, Linz, Austria, August 31 - September 2, 2009. Proceedings*, V. Mařík, T. Strasser, and A. Zoitl, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 13–24.

[11] J. Fitzgerald, K. Pierce, and P. G. Larsen, "Co-modelling and co-simulation in the engineering of systems of cyber-physical systems," in *Proceedings of the 9th International Conference on System of Systems Engineering (SoSE 2014): 9-13 June 2014, Stamford Grand, Glenelg, Australia*, [Piscataway, N.J.]: IEEE, 2014, pp. 67–72.

[12] S. Mittal, J. L. Risco-Martin, and B. P. Zeigler, "DEVS/SOA: A cross-platform framework for net-centric modeling and simulation in DEVS unified process," *SIMULATION*, vol. 85, no. 7, pp. 419–450, 2009.

[13] C.-G. Grundig, *Fabrikplanung: Planungssystematik, Methoden, Anwendungen,* 4th ed. München [u.a.]: Hanser, 2012.

[14] D. Wu, D. W. Rosen, L. Wang, and D. Schaefer, "Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation," *Computer-Aided Design*, vol. 59, pp. 1–14, 2015.

[15] L. Lewis and M. Fowle, *Microservices: a definition of this new architectural term.* [Online] Available: https://martinfowler.com/articles/microservices.html#footnote-etymology. Accessed on: Feb. 08 2017.

[16] M. Richards, "Microservices vs. Service- Oriented Architecture," 2015.

[17] W. Günthner and M. Hompel, *Internet der Dinge in der Intralogistik*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010.

[18] M. Johns, *Getting Started with Hazelcast*: Packt Publishing Ltd, 2015.

[19] O. Kipouridis, M. Roidl, W. A. Günthner, and M. ten Hompel, "Cloud-Based Platform for Collaborative Design of Decentralized Controlled Material Flow Systems in Facility Logistics," in *Proceedings of the 4th International Conference LDIC, 2014 Bremen, Germany*: Springer International Publishing; Imprint; Springer, 2016, pp. 313–322.

[20] S. Mayer, *Development of a completely decentralized control system for modular continuous conveyors*: GRIN Publishing, 2011.

[21] Arizona Center for Integrative Modeling and Simulation, *DEVS-Suite.* [Online] Available: http://acims.asu.edu/software/devs-suite/. Accessed on: Feb. 09 2017.