Increasing the Adaptability of Manufacturing Systems by using Data-centric Communication

Nadine Keddis, Jonathan Burdalo, Gerd Kainz, and Alois Zoitl
fortiss GmbH
Guerickestr. 25
80805 Munich, Germany
{keddis, burdalo, kainz, zoitl}@fortiss.org

***Abstract*—Future manufacturing systems have to be more adaptable to be able to compete in fast changing markets and address specific customer demands. To increase adaptability of production systems, the software and communication infrastructures have to be adaptable as well. Current communication paradigms tightly couple manufacturing systems to communication infrastructures. Changes in the manufacturing system require manual reconfiguration of software and communication infrastructure. In this paper we define the requirements for future adaptable manufacturing systems. They have to be loosely coupled, dynamic, adaptable, and capable of plug & play. We also propose a data-centric approach to make communication systems more adaptable. Using a simplified industrial setup, we show that the approach is feasible for manufacturing systems and increases the adaptability.**

I. Introduction

Traditional manufacturing systems are designed to fulfill dedicated tasks. They are configured accordingly and this configuration is not changed during the lifetime of the manufacturing system. Future manufacturing systems need to be able to cope with volatile markets, shorter product life cycles, and customized products. There are dynamic requirements regarding lot sizes, product variants, lead times, and cost. Adaptable manufacturing systems are necessary to manage these requirements. They have to be flexible enough to quickly change between different products while keeping setup times at a minimum.

Current approaches to increase adaptability focus on mechatronic compatibility. However, introducing a change in current manufacturing processes and rearranging the factory still requires a lot of manual effort and reconfiguration of IT and communication systems. This is costly and time consuming – if at all possible. Costs for setup and installation in current manufacturing systems add up to a third of the total cost [1]. When a change in the production system is necessary, the installation costs and downtimes are even considerably higher. Reasons for this are inflexible communication infrastructures as well as the lack of portability of software components [1]. To have real adaptable manufacturing systems, the IT systems have to be modularized and quickly adapt to changes [2]. The flexibility of IT and communication systems has to be increased. Configuration effort and changeover times have to be kept at a minimum for adding, changing, and removing components. Ideally, all setups should be supported without reconfiguration of IT and communication systems.

Currently, there is no accepted standard for IT interfaces and modules that enables communication within all levels of a manufacturing system. Thus, it is very difficult to add components from different vendors and integrate them with each other. The exchange of information between components is also a challenge in current manufacturing systems. However, as IT is becoming increasingly important for manufacturing [3], new concepts from the IT domain are gradually introduced to the automation industry. A standard software architecture featuring a modular construction and component layout would enable a fast and inexpensive reconfiguration of production lines without the need for high technical expertise.

This paper describes the requirements for the communication and IT infrastructure for adaptable manufacturing systems. We suggest using data-centric communication together with a plug & play capable infrastructure to facilitate the integration of different components and provide a loose coupling of components. This approach aims at standardizing communication mechanisms on all levels of a manufacturing system from the shop floor to the management level. The same infrastructure can be used for horizontal and vertical communication. We focus on showing how the requirements can be satisfied with the data-centric paradigm.

The paper is structured as follows: In Section III paradigms for communication systems are explained. The requirements on communication for adaptable manufacturing systems are stated in Section IV. The data-centric communication paradigm and its benefits for future manufacturing systems are explained in Section V. One example of a data-centric communication infrastructure is presented there as well. Section VI describes the experimental setup and gives an overview of different application areas for this approach. In Section VII the approach is evaluated and discussed. Section II evaluates existing approaches and gives an overview on related work. Finally, Section VIII concludes the paper.

II. Related Work

There has been a lot of work in the field of adaptable manufacturing systems. The adaptability of software and communication systems in the manufacturing industry has also received increased attention in the past years. Much research has been done on transferring service oriented architectures (SOA) from the business domain to the manufacturing domain.

Jammes and Smit state the opportunities and challenges of using SOA in manufacturing [1]. They propose an approach

for integrating SOA in a manufacturing system. However, the approach is based on Ethernet technologies and is not suitable for some legacy systems.

The approach of *Kirkham et al.* looks at applying web services at factory element level to enable a SOA in manufacturing [4]. Their approach helps to bring the business logic closer to the shop floor. Nevertheless, the approach still has some issues that involve scalability problems and quality of service definitions for the interaction.

Ollinger et al. use SOA to ease integration of components [5]. The advantage of using SOA in this case is the ability to reuse control programs. It facilitates programming whenever changes occur. Still, the approach does not consider agility during run-time of the manufacturing system. It does not address adding, removing, and/or replacing components over the life cycle of a manufacturing system while it is operating.

Mendes et al. analyze the usage of SOA in manufacturing in terms of strengths and weaknesses [6]. The main benefits of SOA are the loose coupling of systems as well as the vertical and horizontal integration. Both aspects can be achieved through data-centric approaches as well. One weakness of SOA is however the description methods of the services, which are unknown to automation systems engineers. This is not necessary for data-centric approaches since only data is modeled and the resources can still be programmed with automation languages like IEC 61131.

Calvo et al. and *Ryll and Ratchev* both suggest means to use the Data Distribution Service (DDS) in manufacturing systems. *Calvo et al.* propose a method to integrate DDS with IEC 61499 [7]. The focus here is only on the field level and not on the whole manufacturing system up to the management level. *Ryll and Ratchev* evaluate the applicability of DDS for manufacturing systems [8]. They also only focus on the field level.

There are some approaches to increase adaptability on communication level. One example is the work of *Reinhart et al.*. However, this approach is limited to Ethernet Networks and only provides the plug&play capability on network level.

Middleware approaches like DDS and OPC UA according to *Sauer and Jasperneite* help increase the adaptability of software at field level [2]. However, the authors also recognize that the problem with these approaches is that interfaces have to be known in advance. Alternatively, they suggest using physical variables for communication. This only addresses the decoupling of software modules from PLCs. Additionally, DDS only focuses on the specification of communication requirements, whereas we suggest to specify fixed communication relationships instead.

In contrast to mentioned work, we define a list of required features of future adaptable manufacturing systems. We present a data-centric solution to fulfill future requirements. Besides addressing the manufacturing system as a whole with its different levels, our proposed approach also includes a suggestion to define a global dictionary for data to overcome the problem of unknown interfaces. Compared to available middlewares that support data-centric communication, CHROMOSOME (XME) focuses on extra-functional requirements to give timing guarantees, which is a key requirement for industrial manufacturing systems.

III. Communication Paradigms

Interoperability in industrial automation has been a difficult topic since its origins due to the large amount of vendors and standards that are available. Several middleware technologies have been developed to allow exchange of data between control devices and higher level systems. In this section we describe the most common technologies.

A. Remote Procedure Call

Remote procedure calls (RPC) are a means for remote execution of procedures or subroutines. The programmer writes code in the same way as for local functions and when an RPC is initiated, a temporary binding is created between the computer and the networked resource that will execute the procedure. The most widespread example of this technology in industrial automation is OLE for Process Control (OPC), especially its Data Access (DA) specification that defines the access to real-time data from control devices. OPC DA is based on Microsoft's COM/DCOM technology, which is not further developed anymore. Traditionally, this has been considered a drawback and has raised security concerns. Additionally, it is even incompatible between different Windows versions which is a further disadvantage [9].

B. Service Oriented Architecture Based on Messages / Web Services

These methods define an exchange of information based on a request-response pattern between loosely coupled components. Compared with the previous approach, it reduces the complexity of identification, discovery, and communication among networked components. Data is exchanged over the network with standards such as XML, SOAP, HTTP, TCP/IP, and UDP. Based on these principles, the OPC Foundation released OPC Unified Architecture (UA), which adds many new features with respect to OPC DA while maintaining interoperability. The new features include platform independence, information models to facilitate the communication, security and reliability, and two communication protocols: UA Binary based on TCP and UA XML based on XML/Web Services that can as well be binary encoded for TCP transport.

C. Data-centric Service Oriented Architecture

Data-centric SOA is an evolution of message-based SOA, where the main focus is on the data to be transmitted rather than on the message. The data is exchanged in real-time through low-level binary protocols in a publish-subscribe manner using a global data space. Nodes subscribe to the data or topics they are interested in and can, as well, publish their own data. Solutions such as DDS [10] or XME [11] are based on this paradigm. Section V gives a more detailed overview of data-centric communication and its use in industrial automation.

IV. Requirements for Adaptable Manufacturing Systems

In a manufacturing system, adaptability is required on all levels ranging from the shop floor level to the management level [2]. IT systems have to be able to cope with different types of hardware and software components. Software and hardware in manufacturing systems is typically diverse and often equipped with proprietary interfaces. In order to be adaptable, IT and communication systems have to enable an easy integration of different components and interfaces. Future manufacturing systems should in addition offer services such as self-description, self-configuration, data acquisition, real-time monitoring, and planning of production [12]. To fully enable plug & play in manufacturing systems, standardized description methods have to be used as well [13]. In our previous work [14] we described how to achieve adaptability using a model-based approach to automatically detect resources and their connections within a factory. In this paper we want to highlight further requirements to achieve adaptability and we focus on one possible solution to improve adaptability on communication and software level.

In the past, the system architecture of manufacturing systems has been very monolithic. There was no clear structure and no modules were used. This resulted in a lot of manual configuration effort to implement any change in the system. In this section we give an overview about required features of future adaptable manufacturing systems.

A. Modularity

The first requirement to increase the adaptability is to modularize manufacturing systems [15]. Modularity should be achieved on both the hardware and software side of the system. There are a lot of approaches that tackle the hardware modularity, however there has not been much work done to modularize software as well. To achieve modularity on software level, functionality has to be encapsulated in software modules. Each module has a standardized interface that allows it to interact with other modules. The manufacturing system's software is then composed of required modules that are integrated with each other to fulfill the task. First steps towards modularizing software are taken with the *Application Composer* from 3S[1].

B. Loose Coupling of Components

In the IT domain there has been a lot of research in the field of communication systems. Especially with the increase of Internet applications, communication paradigms for distributed systems have been developed in the past years. In order to cope with a large number of components the components need to be loosely coupled. Loosely coupled components or subsystems have little or no knowledge of the other components in the system. Each of the components has its own encapsulated functionality and is separated from the other systems. Therefore, loose coupling is only possible when modularity is present. The systems interact via a service-based data exchange that requires a semantic description of data. This can be done through direct access to the interfaces of the component, the invocation of services as in SOA, or by specifying the required and provided data as in data-centric communication. Direct communication between components generates very static applications that cannot be changed easily [16]. According to *Eugster et al.* [16], decoupling can be decomposed along three dimensions: space, time, and flow. Space decoupling means that components or subsystems have no knowledge of each other. Components can interact through data exchange, but they do not need to know which component is producing or receiving the data. Time decoupled systems do not have to participate in an interaction at the same time. Components are not blocked until the interaction finishes as in synchronous communication and interaction is mainly asynchronous. Flow decoupling means that the communication activities do not block the main control flow of the components. Communication is done concurrently to other executions so that the main tasks of a component are not blocked. This increases the scalability of the system, since there is no direct dependency between the components. Coordination and synchronization efforts are also decreased when systems are loosely coupled [16].

C. Heterogeneity

Manufacturing systems consist of several types of components ranging from sensors and actuators to databases and high performance computers that need to be integrated. Future manufacturing systems have to include different types of hardware and software as well as available communication infrastructures. It is not feasible to force the use of a specific hardware or software. Future infrastructures must allow the use of multi-vendor and multi-purpose hardware and software while supporting proprietary legacy systems [17].

D. Interoperability and Standardized Interfaces

Because of the large number of different systems available within a manufacturing system, interoperability is a key feature in future adaptable systems. Information exchange must be possible between components from different vendors as well as components for different purposes. Control devices on the shop floor for example must be able to exchange information among each other as well as with planning and monitoring systems on management level using the same mechanisms. Interoperability can be achieved through standardized interfaces and a semantic description of the exchanged data. Standardized interfaces are an important aspect to achieve adaptability. For each component in the system, the interfaces have to be clearly defined and accessible to other components in the system. The interfaces have to be designed in such a way that information for controlling, planning, and monitoring the system is accessible. Without standardization adaptability of software components is quite difficult to achieve [2]. For standardization more effort has to be done especially in creating a common global definition of exchanged data with their semantics and syntax that different developers work with. This has to be achieved through standardization work with several groups and companies. This requirement is stated here for the sake of completeness. However, suggestions on achieving this are beyond the scope of this paper.

E. Scalability

Since future manufacturing systems change frequently, the IT and communication infrastructure has to be scalable to cope

[1]Application Composer: http://www.codesys.com/products/codesys-engineering/application-composer.html

with different setups. The different setups result from either expanding existing systems or reducing them. The system should be efficient with a small number of components as well as with a large number of components. Adding new components to the system should not result in an increase in overhead and added complexity. Performance should not be affected when the setup of the manufacturing system changes. Moreover, the system should not be too complex from the beginning when it only consists of a small number of components.

F. Plug & Play

In adaptable manufacturing systems, control applications of the production and factory setups change frequently. In order to minimize configuration and setup efforts, the control application should be easily exchanged and reconfigured at run-time. Adding and removing components should also be possible with minimum effort. Plug & play provides the capability to connect and disconnect components at run-time. Available components and their description are distributed over the system and thus, components can be used at run-time without further configuration. However, the infrastructure must be designed to enable plug & play. Plug & play is a general requirement for all middleware solutions that can offer reconfigurability, which is necessary for adaptable manufacturing systems [18]. Plug & play is also required on hardware/system level to enable adding new hardware as well. This can be achieved by modeling capabilities of resources as described in previous work [14].

G. Dynamism

The infrastructure of IT and communication systems have to be maintained over the life cycle of a manufacturing system. The life cycle in adaptable manufacturing systems is very dynamic. Dynamism is also triggered by changes in demand, variety of products, and fast changing technologies. Therefore, the infrastructure has to facilitate addition, removal, and replacement of components without affecting the state of the system. In contrast to plug & play, the changes here do not have to occur while the factory is running. They can for example take place during maintenance intervals as well.

V. Data-centric Communication

Current approaches for communication and IT infrastructure are very inflexible because the exchange of data is tightly integrated with the control programs. Any change results in reprogramming the application control program, which involves a lot of manual effort and high costs. To overcome all the challenges and meet the requirements stated in Section IV, we suggest a data-centric communication approach. In this section we give an overview of the data-centric approach and the need for common data semantics to make this approach feasible. Finally, we provide an example of a data-centric middleware.

A. Data-centric Communication

The idea behind data-centric communication is that components communicate with each other using only data as the main means for interaction. The data is decoupled from senders and receivers. Components express their interest in certain data as well as which data they can provide. Based on this, the data is disseminated through the system to the corresponding components. This means that the data flow between software components is not defined explicitly, but is rather specified in terms of publishing and subscribing components. Publishing components of a certain data type provide this data type while subscribing components of certain data types require this data type. Data of the same data type is grouped in a topic. In addition to the data types, publishers and subscribers can add meta information to the data to further refine them. Examples for meta information are time stamps, quality information like accuracy of values, and location information. Meta information is added to the data on demand depending on the configuration of the infrastructure chosen by the user. All communication is done through publications and subscriptions. Other interaction mechanisms should be avoided due to safety reasons. An example of data-centric communication is shown in Figure 1.

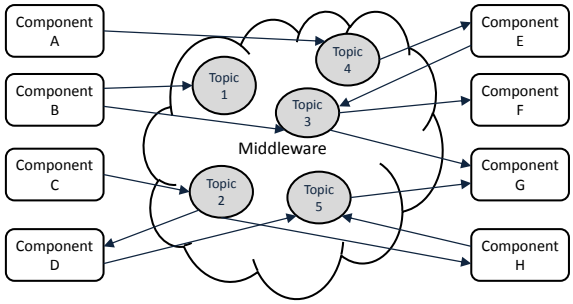

Fig. 1. Components interact with each other by publishing and subscribing different topics.

This approach abstracts from communication technologies and can therefore be used with different protocols and communication hardware. The only restriction is to implement the support for the corresponding operating system running on the hardware and the used communication protocol. Therefore, this approach is suitable for heterogeneous platforms and can always be extended to support new hardware and technologies. Since it can be built on top of other communication systems, it can be integrated in existing systems without further hardware costs as well.

Interoperability is provided because the complete interaction is based on the exchanged data. Nevertheless, it is important to have common data semantics. When the components share the same understanding of the data, applications can interact correctly.

The scalability of the approach depends on how data is disseminated through the system. The system works in the same way regardless of the number of participating components, which is a feature of classic systems as well. However, if the number of components in the system is high, efficient mechanisms for propagating the data and notifying components about available data are necessary.

Dynamism is given when using data-centric communication because components are modular and have an encapsulated functionality. Whenever a component needs to be replaced, the replacing component only has to implement equivalent interfaces for publications and subscriptions. The replacement in this case only affects the replaced component and has no side effects on the whole system. Adding new components is also possible without side effects because the

new communication channels can be automatically established without manual adaptation of old ones. The same applies for removal of components.

Plug & play requirements cannot be fully addressed with data-centric approaches only. However, data-centric communication enables it by adding extra functionality to handle adding and removing of components. Announcing new components can be done in a data-centric manner and the system is then configured accordingly.

B. Common Interpretation of Data

To enable full interoperability with data-centric communication, a global understanding of the exchanged data is necessary. A uniform syntax and semantics of the data has to be available. This can be achieved by describing data independent of specific applications using common descriptions within a domain. A domain-specific data model is necessary to allow interaction between different applications that have no previous knowledge of each other. Currently, developers only specify data required in their application independent from others. This can lead to incorrect applications when systems interact. For example, if two applications need a temperature value, but the format of the data is different, then they are not compatible. The same applies if one application expects the outside temperature, but the other application provides the room temperature. To avoid this, a global domain-specific data model is useful. The global data model can later be modified to also include data models from other domains. This will help achieve the vision of cyber-physical systems where components from different domains can interact with each other easily as well.

C. CHROMOSOME

XME [11] is an open source middleware and platform independent run-time environment. The middleware is developed for distributed embedded systems and PCs. It provides an infrastructure for data-centric communication by implementing a publish/subscribe mechanism. The middleware is independent of the application. It allows the user to focus on developing application logic rather than implementing the infrastructure for it. Based on the definition of publications and subscriptions in the application, the system can configure corresponding communication channels. It has a modular architecture with some core components that have to be available in all configurations. Other components can be added and removed as desired by the user. The architecture of XME is depicted in Figure 2.

XME offers the following features that meet the requirements of Section IV [19]: space, time, and flow decoupling of components is achieved due to the data-centric communication paradigm. It offers flexibility, self-adaptability, and plug & play capability. Different platforms are supported due to the platform independent run-time environment based on a platform abstraction layer.

To enable plug & play, modularity and encapsulation of functionality is necessary. In XME, functionality is encapsulated in software components. Components provide a specific function and interact with other components through defined interfaces (ports). Ports are either publishing or subscribing

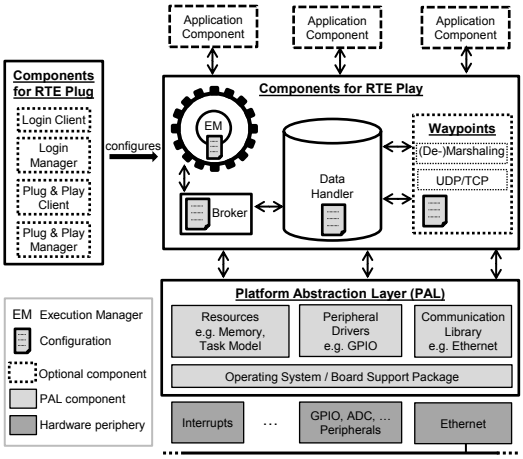

Fig. 2. Architecture of XME [11].

ports that are used to send and receive data respectively. These components can later be reused or replaced without much effort. XME offers a *Plug & Play Manager* that coordinates plug & play processes. This process is divided into a plug-phase and a play-phase. In the plug-phase new components can be plugged into the system. The *Plug & Play Manager* calculates a valid configuration to integrate the new components while the system is running. Once a valid configuration is found, new communication channels are configured accordingly and the system can run with the new components. XME also offers a log-in mechanism to enable adding and removing new devices to and from the system without having to reconfigure the middleware. Each device has a *Login Client* to be able to log into the system. The *Login Client* establishes the communication to the *Login Manager* which is part of the XME ecosystem and enables the plug & play process.

To ensure platform independency and support heterogeneous networks and devices, *waypoints* are used in XME. Waypoints are used for marshaling and demarshaling of data (i.e., converting data to common byte data and back) as well as adding checksums for error detection. There are predefined rules to select suitable waypoints in XME.

Furthermore, XME is designed with a focus on extra-functional requirements (e.g., timing guarantees). All configurations must meet the respective requirements. New components are only integrated into the system if they do not violate the requirements.

In XME, the common interpretation of data is achieved through dictionaries. The dictionaries are specific to a domain and contain a description of all the data that is exchanged with their syntax and semantics. Each of these data descriptions is a topic. Components can then publish and/or subscribe topics. This is necessary to achieve interoperability. Figure 1 illustrates the communication based on topics. To fully achieve interoperability, users from one domain have to agree on required data and their format to specify a usable dictionary for this domain. This can only be achieved if there is collaboration between different users and a standardization of data and dictionaries.

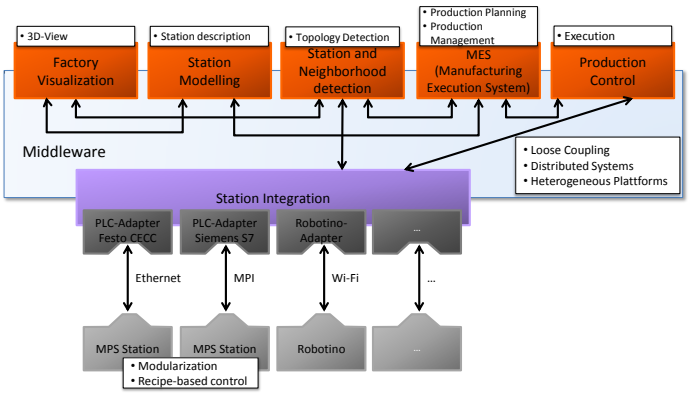

Fig. 3. Horizontal and vertical communication between loosely coupled components using a middleware.

VI. Application Areas of Data-centric Communication

Data-centric communication can be used on many levels within a manufacturing system. On the one hand, it is suitable for integrating business applications on the management level with field-devices on the shop floor level (vertical communication). On the other hand, it is also suitable for communication between devices of the same level (horizontal communication). Since the same communication mechanisms can be used throughout the manufacturing system, the effort to setup and maintain the communication infrastructure can be reduced using the data-centric paradigm. Another advantage of using data-centric communication is that the data can also be sent to visualizing and simulating components without having to add other interfaces and change the configuration. In this section, we give a few examples on how to apply data-centric communication in manufacturing systems.

A. Example Setup

To illustrate the approach, we use a simplified example from the automation domain used for educational purposes, which is a Festo modular production system. The setup consists of different machines that can be arbitrarily combined to form the desired manufacturing system. However, some combinations do not result in valid setups. The manufacturing system can produce black, red, and silver thermometers as well as red boxes. Transporting raw material within the system is also possible. Not all factory setups support all products. The different steps required for such productions include distributing of material, testing, processing, assembly, storage, and delivery. Each of the machines is controlled by a programmable logic controller (PLC) that performs its production steps.

The communication infrastructure is provided by XME. The architecture of the example setup is depicted in Figure 3. The middleware is used to abstract from different components used in the setup. The communication between the shop floor and the management level is done in a data-centric manner.

B. Vertical Integration

Vertical integration involves the integration of sensors and actuators, field devices like PLCs, supervisory control and data acquisition (SCADA) systems, manufacturing execution systems (MES), as well as business applications like enterprise resource planning (ERP) systems. These components have different computational capacities and communication requirements. In current systems, the communication between different levels within manufacturing systems is not possible with the same technologies that are used within each level. Current systems on management level, like MES, also face challenges in adapting to the dynamic and complex shop floor [20]. When data-centric communication is used, the different communication requirements of the different levels like data volume, data rates or timing constraints can be specified and met by the system [8].

We use a data-centric approach to gather information about available resources in the factory to create a model of the factory. Furthermore, information about the topology of the factory is gathered. The topology information is the information about which resources are connected to each other. To create the factory model, the devices on the shop floor level send a signal to the MES containing information about their unique identification and topology information. The MES generates a factory model at run-time and uses it to plan and schedule the production on the respective resources. For this we model each resource that can be part of the factory beforehand using the Eclipse Modeling Framework (EMF). Using a data-centric communication between the resources and the MES we can then generate a model @ run-time of the factory out of the previously modeled resources. Whenever a change in the factory occurs, a notification is sent via the established communication channels and the run-time model can be updated accordingly. In this case each resource in the system publishes to a topic *Add Resource*. The MES subscribes the topic and corresponding communication channels between the different resources and the MES are configured by the system. Resources can also send production statuses to the manufacturing system indicating for example whether a production step is finished or problems have occurred. The vertical communication is also used to control the production. The MES sends commands to the resources to start and stop the production. It can send parameters for the production as well.

C. Horizontal Integration

In manufacturing systems, components within one level are very heterogeneous and have to be integrated. This horizontal integration requires a lot of manual effort. Using a data-centric

paradigm can ease the whole setup effort for a manufacturing system. Different data can be exchanged through the defined topics. If a new device is added, only the new device must be programmed to implement the publications and subscriptions to topics. The other devices in the system remain unchanged. For example, in our setup we exchange signals between different resources to gather topology information. The graphical user interface (GUI) and the planning component of the MES also exchange data through the topics *Start Planning*, *Schedule Generated*, and *Production Finished*. The planning component is responsible for selecting resources for a specific production order and generating a valid schedule for this production order. It also monitors the production progress to tell resources when to start with new orders. The planning is triggered by the user after an order has been placed. A signal is then sent through the *Start Planning* topic to inform the MES about the products to be produced and to trigger the planning. After finishing the planning, the MES informs the GUI about the generated schedules through the *Schedule Generated* topic and the GUI can then visualize them. Through the *Production Finished* topic, the MES can inform the GUI that the current product has been produced so that the GUI can be updated accordingly. The planning component has no real-time requirements; however the resources in the factory are real-time systems. Therefore, the communication between the resources in the factory must fulfill real-time requirements, whereas the communication between the GUI and the MES is only soft real-time. Different requirements on communication can be specified through quality of service requirements for communication relationships.

D. Visualization

Using the same topics as for controlling the real manufacturing system, data from the shop floor can be used to visualize the plant as well as its state. The gathered information about available resources and their topology are sent to the visualization component if it subscribes to the topic and can be used to show the different resources. This mechanism can also be used to validate generated plans and schedules on the simulated resources instead of directly evaluating them on the real hardware. Since the interfaces and communication technologies in the simulated and visualized systems are the same as the real system, there is no manual effort to switch between the two. The automatically generated visualization of our setup is shown in Figure 4. The CIROS Studio tool[2] is used for the visualization. It contains CAD-models of the used machines. The visualization component just subscribes to the *Add Resource* topic and automatically receives the same data as the MES.

E. Integration of Legacy Systems

In order to integrate legacy systems a gateway can be implemented to connect the legacy system to the data-centric system. The gateway implements communication protocols of the legacy system and at the same time communicates in a data-centric manner with the rest of the system. Because of the large number of available technologies, a trade-off between the functionality and the universality of the gateway has to be made [21].

[2]CIROS: http://www.ciros-engineering.com/home/

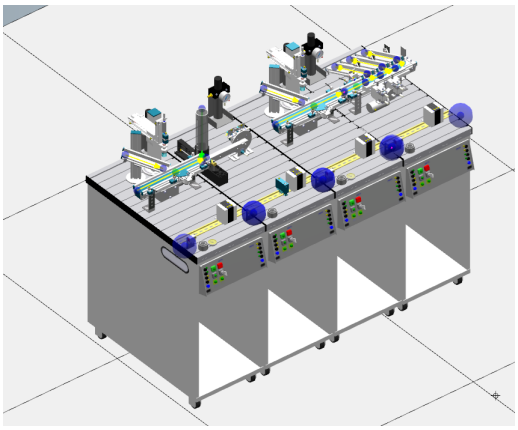

Fig. 4. An automatically generated visualization of the system.

VII. Discussion

In this section we evaluate the data-centric approach for adaptable manufacturing systems.

A. Benefits

The data-centric communication seems to be suitable for increasing the adaptability of manufacturing systems. It provides a loose coupling of components. The data-centric paradigm has received a lot of attention in the past years also in large scale applications over the Internet and is providing a necessary decoupling for the distributed applications there. It is suitable for heterogeneous platforms where different types of devices have to be integrated. Legacy systems can also be supported because the paradigm can be implemented on top of available infrastructures. For proprietary systems, gateways can be implemented. Adding and removing components over the life cycle of the manufacturing system is easier than before because new components only have to specify the data they are interested in and the communication channels are automatically configured accordingly. The same applies for replacement of components. This meets the requirement of being dynamic.

Another benefit of the data-centric approach is the support of vertical and horizontal integration of devices in the manufacturing system. The business logic gets closer to the shop floor. This also helps to establish the manufacturing system as one whole system and not as a collection of isolated solutions.

The communication paradigm can be easily extended to support plug & play. CHROMOSOME is one example of a middleware that uses a data-centric communication paradigm while enabling plug & play.

Since the data-centric approach is not restricted to one domain, it is also useful to connect other domains to the manufacturing domain to achieve the vision of the factory of the future [12]. The same communication paradigm can be used to connect the factory directly to the retailers where customers can directly order their products and send the desired requirements to the shop floor. Another application scenario for the future would connect manufacturing systems to smart grids to manage energy more efficiently.

B. Issues

To achieve interoperability with this approach there is a strong need to standardize interfaces and data models. Without standardization a full adaptability on software level is not possible [2]. Extending standardization efforts to include other domains can result in defining global dictionaries with sub-dictionaries for each domain.

There are few frameworks and middlewares available that are based on a data-centric approach and are suitable for the manufacturing domain. The most famous standard from the IT domain is DDS. There are some implementations based on the DDS standard. However, available frameworks have to be adapted to meet the requirements of manufacturing systems. There is a need to evaluate different implementations of the data-centric paradigm to find a suitable solution for the automation industry.

Data-centric communication is still not often used within manufacturing systems and there is still some work that needs to be done to make it usable in real industrial setups. Nevertheless, the experimental setup showed that the approach is feasible and that further improvements have the potential to make it usable in industrial setups as well.

VIII. Conclusion

In this paper, we explained the requirements of future manufacturing systems which have to be adaptable to face turbulent markets. The key features of future manufacturing systems to increase adaptability are decoupling of components, support of heterogeneous platforms, scalability, Plug & Play, and dynamism. Additionally, standardized interfaces are necessary which are closely related to having interoperability. We proposed to use a data-centric communication approach to address these challenges. This approach results in a decoupled system that involves heterogeneous devices because the communication paradigm is not restricted to certain devices and technologies. It is also very dynamic because components only specify their communication needs in form of publications and subscriptions of data. This makes it quite easy to add, remove, and replace components along the life-cycle of a manufacturing system. Plug & Play can be added to the data-centric approach as showcased by the CHROMOSOME middleware. It is used in the simplified industrial setup that was used to evaluate the approach. The example setup confirmed that a data-centric communication paradigm meets the requirements of adaptable manufacturing systems and helps decoupling components in time, space, and flow. However, there is still the need to evaluate different implementations of the data-centric approach and further specify constraints to find a suitable framework for manufacturing systems.

References

[1] F. Jammes and H. Smit, "Service-Oriented Paradigms in Industrial Automation," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 62–70, 2005.

[2] O. Sauer and J. Jasperneite, "Adaptive information technology in manufacturing," in *CIRP Conference on Manufacturing Systems*, Madison in Wisconsin, USA, Jun 2011.

[3] B. Vogel-Heuser, G. Kegel, K. Bender, and K. Wucherer, "Global Information Architecture for Industrial Automation," *atp*, 2009.

[4] T. Kirkham, D. Savio, H. Smit, R. Harrison, R. Monfared, and P. Phaithoonbuathong, "SOA middleware and automation: Services, applications and architectures," in *6th IEEE International Conference on Industrial Informatics (INDIN), 2008*. IEEE, 2008, pp. 1419–1424.

[5] L. Ollinger, J. Schlick, and S. Hodek, "Leveraging the agility of manufacturing chains by combining process-oriented production planning and service-oriented manufacturing automation," in *Proceedings of the 18th IFAC World Congress*, 2011.

[6] J. M. Mendes, P. Leitao, and A. W. Colombo, "Service-oriented computing in manufacturing automation: A SWOT analysis," in *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 346–351.

[7] I. Calvo, F. Perez, I. Etxeberria, and G. Moran, "Control communications with DDS using IEC61499 Service Interface Function Blocks," in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2010, pp. 1–4.

[8] M. Ryll and S. Ratchev, "Application of the Data Distribution Service for Flexible Manufacturing Automation," *International Journal of Mechanical, Industrial and Aerospace Engineering*, vol. 2, p. 3, 2008.

[9] W. Granzer and W. Kastner, "Information Modeling in Heterogeneous Building Automation Systems," in *9th IEEE International Workshop on Factory Communication Systems (WFCS), 2012*. IEEE, 2012, pp. 291–300.

[10] Object Management Group. (2007, January) Data Distribution Service for real-time systems. http://www.omg.org/spec/DDS/.

[11] C. Buckl, M. Geisinger, D. Gulati, and F. J. Ruiz-Bertol, "CHROMOSOME: A run-time environment for plug & play-capable embedded real-time systems," in *6th International Workshop on Adaptive and Reconfigurable Embedded Systems (APRES 2014)*. ACM, Apr. 2014.

[12] *Cyber-Physical Systems: Driving Force for Innovation in Mobility, Health, Energy and Production*, ser. acatech Position. acatech – National Academy of Science and Engineering, 2011.

[13] O. Sauer and M. Ebel, "Plug-and-work von produktionsanlagen und übergeordneter software," *Aktuelle Trends in der Softwareforschung, Tagungsband zum do it. Software-Forschungstag*, pp. 24–33, 2007.

[14] N. Keddis, G. Kainz, C. Buckl, and A. Knoll, "Towards Adaptable Manufacturing Systems," in *IEEE International Conference on Industrial Technology (ICIT), 2013*. IEEE, 2013, pp. 1410–1415.

[15] H.-P. Wiendahl, H. ElMaraghy, P. Nyhuis, M.-F. Zäh, H.-H. Wiendahl, N. Duffie, and M. Brieke, "Changeable Manufacturing – Classification, Design and Operation," *CIRP Annals-Manufacturing Technology*, vol. 56, no. 2, pp. 783–809, 2007.

[16] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[17] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina, "A real-time service-oriented architecture for industrial automation," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 267–277, 2009.

[18] R. Marau, L. Almeida, M. Sousa, and P. Pedreiras, "A middleware to support dynamic reconfiguration of real-time networks," in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2010, pp. 1–10.

[19] S. Sommer, M. Geisinger, C. Buckl, G. Bauer, and A. Knoll, "Reconfigurable Industrial Process Monitoring using the CHROMOSOME Middleware," in *The Fifth International Workshop on Adaptive and Reconfigurable Embedded Systems (APRES 2013)*. ACM, April 2013.

[20] A. Bratukhin and T. Sauter, "Distribution of MES Functionalities for Flexible Automation," in *In Proceedings of the 8th IEEE International Workshop on Factory Communication Systems (WFCS), 2010*, J. P. A. Willig, Ed., Nancy, France, May 2010, pp. 157–160.

[21] T. Sauter and M. Lobashov, "How to Access Factory Floor Information Using Internet Technologies and Gateways," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 699–712, 2011.