A Service-Oriented Approach for Increasing Flexibility in Manufacturing

Christin Groba
Technische Universität Dresden
Institute for Systems Architecture
Computer Networks Group
christin.groba@tu-dresden.de

Iris Braun
Technische Universität Dresden
Institute for Systems Architecture
Computer Networks Group
iris.braun@tu-dresden.de

Thomas Springer
Technische Universität Dresden
Institute for Systems Architecture
Computer Networks Group
thomas.springer@tu-dresden.de

Martin Wollschlaeger
Technische Universität Dresden
Institute of Applied Computer Science
Industrial Communications Group
martin.wollschlaeger@tu-dresden.de

Abstract

Manufacturing environments are characterized by a multitude of heterogeneous devices, networks, specific protocols and applications. Therefore, static structures, close coupling and vendor-specific solutions have been established over decades. To increase flexibility and interoperability we introduce a service-oriented middleware approach in form of an integration layer between shop-floor equipment and enterprise applications. The integration layer addresses the actual capabilities and data flows on the field and control layer individually. It offers core services that are applicable in several manufacturing domains. We describe a detailed case study to discuss pros and cons of a SOA approach in manufacturing systems and to show the feasibility of our approach.

1 Introduction

Manufacturing environments are characterized by a layered system structure containing a variety of heterogeneous devices, networks, protocols and applications. Although actual developments are focusing on integration aspects between shop-floor systems and enterprise applications, there is still a lack of standardized interfaces between these two worlds. SOA seems to be a promising approach to bridge this gap by achieving interoperability between automation devices and shop-floor applications on the one hand and enterprise applications on the other hand. The challenges SOA approaches face is handling different solutions from different vendors as well as the diversity of network protocols like Industrial Ethernet (e.g. PROFINET, EtherCat, Ethernet/IP, Ethernet Powerlink) or non-IP-based protocols like fieldbusses according to IEC 61158 [1] (e.g. PROFIBUS, CAN, DeviceNet, AS-Interface). Even if technological development may promise a reduction of this variety, heterogeneity resulting from different application areas, and from different industry sectors is rather increasing. Thus, flexible, reliable and easily adaptable middleware solutions are required by industry.

The paper aims at demonstrating and discussing the feasibility of a SOA-based middleware approach in the field of manufacturing systems. Based on a case study about maintenance operation management, we show benefits of a service-oriented middleware layer for integrating heterogeneous system structures in general, and shop-floor equipment and enterprise applications in particular. We describe basic concepts and the architecture of our service-oriented middleware approach before presenting implementation details for our particular use case. The discussion of design and implementation decisions regarding service and interface granularity, architecture and core services as well as communication aspects and data handling points out the generality of our approach regarding the manufacturing domain.

The paper is organized as follows: According to a case study chances and challenges of SOA in manufacturing are discussed in section 2. Section 3 covers related work in this realm of research. Basic concepts and the architecture of the proposed integration layer are introduced in section 4. Section 5 examines implementation considerations. Section 6 summarizes the results and cognitions providing an outlook on future tasks afterwards.

2 SOA in manufacturing

The application of the service-oriented paradigm in office and telecommunication scenarios is frequently discussed by researchers and industry experts. Services encapsulate pieces of functionality in such a transparent way, that they can be described, published, located and dynamically invoked in a programming language and system independent way. However, to what extent are services feasible off the beaten track, in application domains like man-

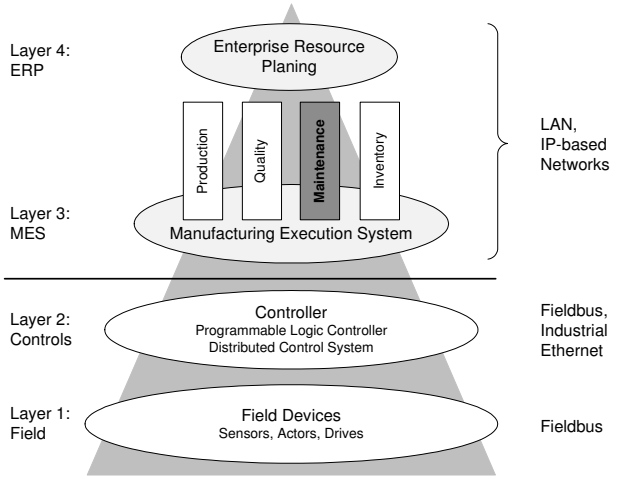

Figure 1. Layered structure of manufacturing systems according to [2]

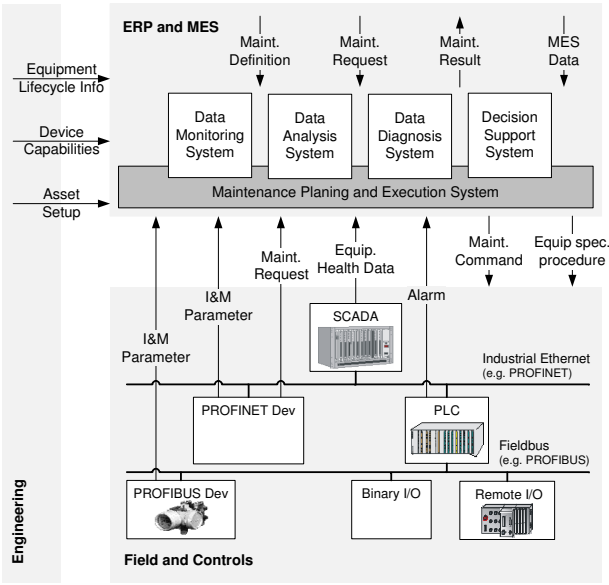

Figure 2. Case study maintenance operation management

ufacturing?

Typical manufacturing systems are organized in a hierarchical structure. Fig.1 shows the functional layers as defined by conceptual standard IEC 62264 [2]. We believe that maintenance operation management in layer 3 is a promising candidate for applying the service-oriented approach in the manufacturing environment. Compared to production operation management, i.e. control of the production process itself, maintenance operation management does not expose real-time requirements. Real-time requirements usually demand a close coupling of system components and make the application of a SOA approach complicated in practice. Moreover, maintenance operations have high potential to improve production processes, especially by avoiding process slowdowns and production downtimes caused by functional loss of manufacturing equipment. Despite of this potential, up to the present maintenance has been rather neglected for the benefit of optimizing production. Thus, maintenance operation management is addressed in the following case study and is further examined for its service capability.

2.1 Case study: maintenance

Functional loss of manufacturing equipment is caused by aging, wear or drift of operating points. In fact, equipment degradation results in expensive process slowdowns, loss of quality, safety shutdowns or even machine and production downtimes. However, an enormous potential for reducing asset costs can be identified in the realm of maintenance because well maintained machines are less likely to fail. Therefore, the following case study will analyze the state-of-the-art maintenance setting which is illustrated in Fig.2.

Strategically, maintenance may be performed *correctively* after machine malfunction has been detected, *preventively* in periodic maintenance intervals or *predictively* based on the current machine state. Thus, maintenance is triggered by feedback of human experts after inspection, time constraints indicating a new maintenance interval, or transgression of predefined threshold limits and condition monitoring. Based on business and operation processes in place, a maintenance request is generated from low-level or high-level activities.

On field and control layer instruments and devices comprise different degrees of intelligence. Sophisticated devices may contain embedded maintenance functions to automatically issue a maintenance request. PROFINET and PROFIBUS devices, for example, implement identification and maintenance (I&M) functions [3] which allow online access to device identity information. Other field devices like simple Binary I/O Modules are less intelligent. To achieve the same quality of information a sequence of queries may has to be performed. However, less capable devices are connected to controllers (PLC) and data acquisition systems (SCADA) that have appropriate resources to derive and forward maintenance alarms as well as equipment health data.

On MES and ERP layer a multitude of maintenance assisting systems process data acquired from the shop floor. Systems responsible for data monitoring, data analysis and diagnosis as well as decision support provide high-level activities to deduce a maintenance request. However, each system is a vendor-specific application varying in functionality, interfaces and data representation. Therefore, upper layers are characterized by heterogeneity and redundancy also. This situation is caused by historical, political and technical reasons. While system concepts grew and changed over the years, mainly supported by

technical achievements, competition between manufacturers prevented more uniform solutions. Finally, the heterogeneity of information consumers namely operators, production scheduling experts, quality experts, maintenance staff, asset optimization teams etc. caused the development of different applications and tools, used in parallel without interconnections.

After a maintenance request has been issued the maintenance planning and execution system schedules the required work orders typically by considering available maintenance resources such as personnel, material and maintenance equipment. However, there is a variety of information that is rather neglected in current maintenance solutions. Data provided by the MES such as future production plans and changeover times could be integrated to setup an advanced maintenance schedule that least interferes with production. Furthermore, the incorporation of information about the equipment's setup, life cycle and capabilities would add additional value to the decision on appropriate maintenance strategies and actions. This information, however, is stored in a distributed and rather closed engineering system which exists in parallel to the ERP, MES and automation system.

In order to fulfill the scheduled maintenance operations, maintenance personnel gets instructed through maintenance definitions such as equipment and system drawings, engineering documentation, specifications, vendor manuals and standard operating procedures for repair and servicing. As already pointed out this information is spread over many data sources, is available in different formats and is accessible via different communication systems and services. The equipment-specific maintenance command e.g., calibration or cleaning command are downloaded to layers 1 and 2. Afterwards, the maintenance result as well as equipment data indicating the equipment's past, current or future health condition is returned. Having executed maintenance work orders by following all defined regulations and procedures, the maintenance planning and execution system collects and analyzes maintenance data to calculate its effectiveness and to identify areas of improvement.

2.2 Challenge: flexibility and interoperability

The challenge of maintaining manufacturing equipment is to cope with heterogeneity on layers 1 and 2 as well as on layers 3 and 4. Field devices tend to offer increased functionality. However, a generic, vendor-independent and complete model of field devices is lacking. Thus, a multitude of control concepts and maintenance tools is the consequence. Moreover, heterogeneity on layers 1 and 2 is present because of the individual complexity of different industry branches e. g. manufacturing and chemical engineering. Furthermore, different data sources each with individual capabilities exist to acquire maintenance relevant data. Data transmission frequencies, transmission volumes as well as data representation depend on the specific process and measured signal. On the other hand, heterogeneity on layers 3 and 4 is exposed by the functional diversity of tools supporting data analysis, reporting, diagnosis as well as maintenance planning and execution. Each tool has individual requirements, dependencies and internal data formats that have to be satisfied. Common to all tools, however, is their reliance on condensed information. Instead of a vast amount of raw data, tools rather process filtered and recombined maintenance information that is enriched with information from other sources.

Analyzing the case study the need for flexibility, integration and interoperability in the manufacturing environment becomes evident. Because heterogeneity does not only exist between lower and upper layers but also in each layer itself, the challenge is to interconnect these layers and to make the resources of one layer accessible to the others. Although integrated solutions already exist (e. g., Siemens TIA), they are vendor-specific and limiting because they lack the flexibility to interconnect multi-vendor systems. Therefore, to achieve real interoperability between systems and layers individual features and requirements have to be encapsulated and appropriately mapped to a common interface. We propose an integration layer between the control and MES layer that hides layer-specific diversity by providing services offering such a common interface. The integration layer will especially address actual capabilities and data flows on lower layers not assuming a certain intelligence of field devices or IP-based communication systems.

3 Related work

OPC (OLE for Process Control) [4] is considered a defacto standard for data exchange between shop-floor automation equipment and higher layers of the automation pyramid. Current activities like the specification of OPC Unified Architecture (UA) [5] are focusing on platform-independent solutions based on Web services. It can be expected that the new OPC UA specification will adopt the relevance of the COM-based variant of OPC. Thus, data exchange between the layers 1/2 and 3/4 will have to consider OPC. However, OPC is still focusing on the pure communication aspect and will, therefore, provide communication services. Related to the aim of providing information exchange between the layers, further definitions are still necessary.

The authors of [6], [7] propose the use of Web services to exchange data between production workflow and production scheduling applications situated in layer 3. Communicating either proprietary or using OPC DA [4] typically parameters, measurement and control values are exchanged.

To integrate information suitable for maintenance tasks into CMMS systems a generic platform for e-maintenance was developed by the PROTEUS project [8]. Applications may use the platform by providing adapters mapping their (mostly proprietary) interfaces to a Web service

based message scheduling system. Layer 2 is accessed via data acquisition servers that adapt specific communication systems to OPC. Apart from communication, no maintenance-related services have been defined. Services as addressed in this paper could be implemented into the adapters.

A standard information storage schema for asset and maintenance information was defined by MIMOSA [9]. Typical applications of that schema reside in layers 3 and 4. Layer 2 is accessed solely through OPC. As an organization, MIMOSA assigns an enterprise a globally-unique, alpha-numeric user tag identifier value. With such a globally unique enterprise ID and an enterprise unique asset identifier, any asset may be transferred to another site or enterprise without causing doubling of identifiers. On basis of this ID, assets (as part of the equipment) can be addressed in order to link additional information e.g. maintenance relevant engineering data.

The SIRENA project [10] develops an infrastructure for high-level communications between devices by utilizing the Device Profile for Web Services (DPWS) [11]. Although DPWS allows seamless integration of new devices and their provided services, this approach is limited to special use cases requiring WS-enabled devices and IP-based communication protocols. However, reality still shows the dominance of proprietary standards and protocols that severely impede the progress towards flexibility and agility. Therefore, our approach builds on common proprietary standards and uses an abstraction layer for realizing interoperability.

A design, execution and management platform for next-generation industrial automation systems is developed by the SOCRADES project [12]. The functionality of smart embedded devices is encapsulated and advertised by a SOCRADES service. Instead of re-programming large monolithic systems, loosely coupled embedded units are reconfigured to achieve increased adaptability. Although, SOCRADES' intention of exploiting the service-oriented architecture paradigm both at device and application level is quite similar to ours, we envision an integration layer that copes with individual capabilities of field devices and communication systems in place, not assuming a certain level of intelligence.

PABADIS'PROMISE [13] introduces a new control architecture to increase adaptability and flexibility of manufacturing systems especially targeting production and supply-chain management. Mobile real-time agent systems and RFID technologies are employed to distribute ERP system building blocks even on field control level. The physical migration of agents stored in RFID tags ensures a close connection between agent and product. Therefore, on-the-fly design of order-related control applications is possible. We differ from the mature PABADIS'PROMISE in such that we envision a service-oriented abstraction layer providing valuable domain-independent core services.

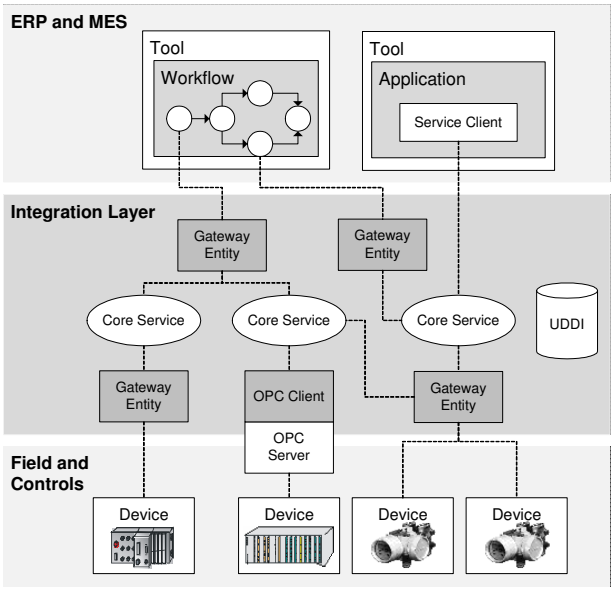

Figure 3. Integration layer architecture

4 Basic concept and architecture

One of the main prerequisites for optimizing the manufacturing environment is to open up new information sources and to provide seamless access to them. However, architecture and communication methods vary greatly. While MES and ERP layer are similar to standard IT architectures and communication methods (i.e. Web services), the field and control layers are realized specifically with Industrial Ethernet and fieldbus systems. Our proposed integration layer will abstract from this heterogeneity by providing services that offer a common interface to the outside world, however, that internally adapts its functionality to the given situation of a certain environment.

4.1 Architecture

To integrate lower layers as information sources, an integration layer illustrated in Fig.3 is designed as middleware between the control and MES layer. The integration layer addresses service granularity, physical distribution and degree of coupling individually according to the actual capabilities and data flows on the field and control layer.

The integration middleware offers new-quality services adding further value to existing tools and applications. Applying the SOA paradigm, services abstract from lower and upper layer heterogeneity and implement valuable functionality that is applicable in several application scenarios. For instance, a core service for device identification is evidently beneficial in production operation management and maintenance operation management as well. Hiding implementation details behind service interfaces, services provide a common access mechanism for above layers to receive information offered by lower layers. In fact, services introduce more flexibility because they dis-

solve fixed, solution-specific point-to-point mappings between upper and lower layers.

Specific gateway entities enable legacy systems to consume core services without being changed. Acting as a proxy they map application-specific requests to core services and from there to sequences of services provided by the underlying systems. To map core services to specific underlying communication protocols, a gateway entity is created for each technology and device type. An example for such a lower gateway entity is the combination of a OPC client and OPC server. Within the entire abstraction process the gateway entity considers domain-specific semantics as well as device and system models. Those models have to be formally described in order to identify information sources and to allow mappings.

A UDDI resides in the integration layer to publish core services and to make them plant-wide accessible. The Universal Description Discovery and Integration (UDDI) protocol was actually designed for companies to quickly, easily, dynamically find and use Web services over the Internet. The UDDI in the integration layer does not address this global business scope. Therefore, registered services are visible only to company-internal MES and ERP systems. In case of maintenance operation management a gateway entity representing a CMMS system may use the UDDI to search for services to better coordinate and equip maintenance personnel carrying out maintenance tasks.

An open topic that remains for future research is to keep the architecture flexible for advanced tools and applications disposing appropriate resources to access core services directly. A service client implemented within such applications would replace a specific gateway entity in the integration layer. This development would finally result in the distribution of integration layer responsibilities.

4.2 Data handling

Information offered by layers 1 and 2 can be distinguished between real-time data, parameter data, and fixed data. Real-time data is time-critical for the automation and control function. Data associated with measurement values, set points, alarms, and controller input and output is typically available at runtime and can be accessed through standardized interfaces like OPC. Parameter data is used to adapt devices to their specific function in the system. Rather rarely exchanged, parameter data is mostly shared with diagnosis and engineering tools. Fixed data contains information on manufacturer, version, etc. and is used in an inventory and asset management context. Because of its rather static nature and low update rate a frequent transmission of fixed data over OPC is not necessary and would waste bandwidth. Therefore, this data is normally accessed by specific communication services. The semantics of data supplied by layers 1 and 2 is defined in so-called profiles (e.g., for different device types and different application scenarios).

4.3 Core services

A feature of the integration layer are its core services. Addressing domain-independent tasks (e.g., device identification) and being applicable in various domains core services raise the benefit of the integration layer. In the following we outline some example core services emphasizing that this list is not exhaustive.

4.3.1 Device identification service

Device identity represents the common reference point for all asset life cycle stages and related device information. However, this identification data is spread over different information sources and may have to be mapped between life cycle stages. Depending on the particular device intelligence certain identification information is available on the field and control layer. Therefore, a *device identification service* will open up this information source by adapting to the situation in place and providing device identification information. Some devices supply identification data themselves. In this case, data can be read out over the communication system. However, due to different communication protocols information has to be accessed in various ways. For example, I&M parameter of a PROFINET device are accessed over PROFINET (industrial ethernet) whereas PROFIBUS devices, in case they offer I&M parameter, are accessed via PROFIBUS (fieldbus). On the other hand, identification information of devices and passive equipment (e.g. pipes and tubes) is sometimes spread over several sources in layer 1 and 2. Therefore, the service relies on specific proxies (gateway entities) that know for a certain device type which information is found where.

4.3.2 Information retrieval service

Devices and systems typically exist in a life cycle characterized by the stages: design and production (at the manufacturer), planning, engineering, run-time, maintenance (at the end user) and recycling. During this life cycle, device-related information is distributed throughout the whole system and hard to retrieve. A maintenance expert may not easily access engineering documents due to separation of concerns and isolated data sources. Or parts of information needed exists within devices, configuration files and planning documents. Thus, an *information retrieval service* will discover device-related information from different contexts and will apply application-specific filters to reveal relevant information. Assembling or even representing all information in a uniform format is challenging because documents such as manuals in PDF format may not be convertible. Therefore, using ontologies that formally describe available documents the service lies focus on information discovery and provides references to it.

4.3.3 Device positioning service

Knowledge about where a device resides is crucial for production and maintenance operation management. Because the position of a device has many aspects, different structure and location services are necessary. In large assets where several factory floors and thousands of devices are managed, it seems cumbersome for maintenance personnel to keep an overview of where machines are physically located. A *device positioning service* will find devices by deriving position information from appropriate sources e.g., GPS coordinates stored in a database are loaded to a handheld and guide a maintenance engineer through sanitation.

4.3.4 Structure analysis services

A device is characterized by its affiliation to several structures e.g. physical network, functional network and device structure. Detecting and representing these would be beneficial to several experts within the life cycle. An engineer may utilize the *physical network service* to determine the network type by which a device is accessed. This service will further infer network-related properties such as physical neighboring devices in fieldbus systems. A *device structure service* will reveal the structure of complex devices by considering the device's planned configuration and detecting installed modules. As soon as the actual configuration deviates from the reference configuration alarms are issued. A production planer is rather interested in functional network of devices. A *functional network service* identifies functional capabilities of a device, filters relevant information for production and even suggests functional identically device alternatives. Further it may determine the functional predecessor and successor concerning a particular process chain, e.g., which sensor affects which actuator in a control loop.

4.3.5 Event notification service

For some applications it might be necessary to stay informed about the actual state of a device or processing step, e.g. to initiate logging of data under certain circumstances. State information, however, is usually mapped to alarms/events and send to a PLC. Because most devices wait on an acknowledgment of the PLC it is hardly possible to access this eventing mechanism. Therefore, an *event notification service* will make use of the communication function blocks offered by the PLC and will subscribe to the PLC for such events. Besides the propagation of events to layers 3 and 4, mapping of proprietary or vendor-specific event codes to generalized ones is necessary. A step into this direction is provided with specifications like [14]. These specifications define common maintenance states and data structures for maintenance as well as mappings to appropriate transport mechanisms for specific protocols (in this case PROFINET I/O).

5 Implementation

A SOA enabling technology are vendor-neutral Web services. Available independently from the implementation of particular applications, three mechanisms are associated with Web services:

- SOAP messages to exchange data between service provider and service consumer
- WSDL to propagate the interfaces a service implements
- UDDI for potential service discovery

Web service are considered an abstraction layer in IT architectures due to their service interfaces that provide a simple and comprehensible access to underlying applications. However, to fulfill the aforementioned integration capabilities, Web services alone are not sufficient. In fact, the implementation platform has to satisfy four primary requirements:

- messaging between core services and applications on different automation layers
- intelligent message routing
- data and message format transformation
- Web service support and discovery

From a technological perspective JBI and ESB address these issues.

Java Business Integration (JBI) [15] is a standardized platform that entails a service-oriented approach for the integration and structuring of enterprise functions in J2EE environments. It creates a standard meta-container allowing several components to be plugged in and to interoperate seamlessly. This framework provides a dynamic composition and deployment of loosely-coupled applications. Regardless of the implementation technology, multiple services may be wrapped as a service unit and packaged in a loosely-coupled manner into a service assembly. Messages exchanged between various service engines like BPEL engines or external Web services, are normalized through binding components. The normalized message router handles message routing. JBI proves its flexibility and resilience by allowing extensions for management, monitoring, and other enhancements. Despite being an open specification, JBI is not widely adopted in the business world. An open source project, pioneered by Sun Microsystems known as Open ESB [16], fully implements JBI for the construction of a powerful ESB.

Enterprise Service Bus (ESB) is a larger integration concept involving intelligent routing, XML transformation, federated management, orchestration and process flow. Offering a distributed middleware infrastructure ESB is consistent with SOA principles. The basic idea of this bus architecture is to link each service or legacy application (wrapped by a service frame) directly to the

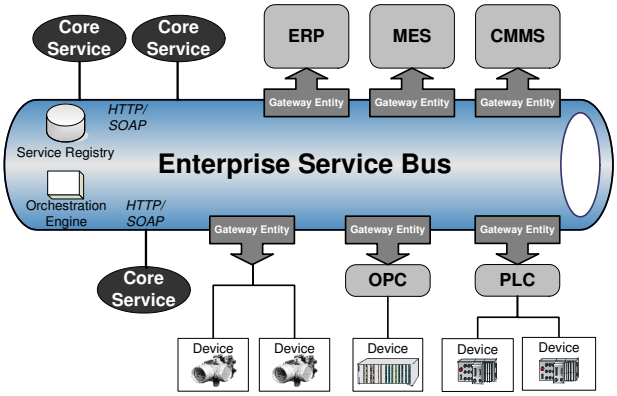

Figure 4. Enterprise Service Bus

bus. Thus, large numbers of point-to-point connections between applications and services are avoided. The find/bind/invoke mechanism for services is no longer implemented within applications or services but rather managed by ESB in combination with configuration and deployment tools. In fact, providing a highly distributed integration environment that extends well beyond hub-and-spoke architectures, ESB facilitates a clear separation of business and integration logic [17]. ESB features value-added services beyond those found in basic communication middleware, such as message validation, transformation, content-based routing, security, service discovery, load balancing, and logging.

We consider ESB as a suitable approach for integrating manufacturing systems (see Fig.4). As a loosely coupled, standards-based, highly distributed integration layer it scales beyond the limits of a hub-and-spoke EAI broker. Combining messaging, Web services, data transformation and intelligent routing an ESB reliably connects and coordinates interaction between a significant number of diverse applications across different automation layers.

Core services, realized as Web services with Apache Axis, directly link to the bus via SOAP and Web service APIs. Because they rely on several data sources (e.g., devices and controls, engineering storage, MES and ERP storage), different gateway entities wrap access to legacy applications.

Gateway entities are service containers that realize the actual integration. They adapt ERP and CMMS applications that cannot directly interact with core services. They integrate specific underlying communication protocols from layer 1 and 2. For each technology and device type an individual gateway entity is created. An example for such a lower gateway entity is the integration of a OPC client connecting to a OPC server. Depending on device intelligence gateway entities further encapsulate different message exchange pattern e.g., they subscribe to a PLC and use the event generation capability of function blocks. ESB implementations like JBossESB [18] already provide a diversity of connectors especially for higher-level applications: They utilize J2EE components such as Java Message Service (JMS) and J2EE Connector Architecture (JCA or J2CA) or nicely integrate with applications built with .NET, COM, C#, and C/C++.

A service registry stores all necessary information about service endpoints the ESB needs to realize the find/bind/invoke mechanism for services. Further, the registry allows to dynamically add and remove core services and gateways. Therefore, the integration layer becomes flexible and easily extensible.

An orchestration engine (e.g., BPEL engine) facilitates the basic concept of ESB: Service are not invoked directly by other services, rather they are part of a larger event-driven process flow. The following example illustrates this. An alarm from a PLC is put as XML message into a topic or queue. On message arrival (entry point) a device identification service starts processing and places the alarm message augmented with identification data into its exit point. Then a subsequent maintenance execution service receives this message via a queue or topic and creates a maintenance command message. This message is then put into the appropriate queue or topic to be received by the PLC. Such a process chain (services, entry and exit point) has to be configured in advance. The ESB realizes the process at runtime by identifying the next service in chain, binding to it and invoking the service.

Direct service access within ESB is also possible, however, should be an exception. In this case the development of dynamic invocation interfaces for remote service invocation is favored. Loose coupling between a service and its client is essential, i.e., a client dynamically accesses a service without changing its core application. Especially, if service endpoints often change or several service instance exist, only the abstract interface should be known during design time while the particular service is dynamically localized at run time. JAX-RPC API allows service discovery at runtime via JAX-R registry. If this API is accessed directly no additional stub classes have to be generated and client-service dependability is reduced. At runtime only the specific port, operation and input parameters have to be supplied. Hence, dynamic invocation interfaces allow flexible client design.

Using ESB is so far a conceptual consideration. Currently, we have exemplary implementations of few core services and gateway entities in the maintenance scenario. We are planning the integration with an ESB within next months.

6 Summary and outlook

Analyzing the maintenance case study, two major findings became evident: Firstly, manufacturing systems are characterized by diversity including heterogeneous devices, networks, protocols and applications. Secondly, although a great amount of relevant data resides in the field and control layer, it is rarely used due to the lack of standardized interfaces between control and MES layer. Therefore, the challenges in manufacturing systems are

bridging the gap between shop-floor and enterprise applications, adapting and encapsulating heterogeneity in place, and leveraging flexibility and interoperability of systems.

By applying the SOA paradigm, the proposed integration layer defines core services to abstract from diversity and provides service interfaces for a common access mechanism. Gateway entities are responsible for the appropriate mapping between the layers. For implementation ESB is favored because it fits the requirements for loose coupling and high distribution of the integration layer.

The benefit of our approach is its adaptation to the current manufacturing system not assuming a certain device intelligence, device resources, or IP-based protocols. Gateway entities encapsulate the integration complexity for a particular device type, communication protocol, or ERP/MES tool. They direct service requests from higher-layer legacy systems to sequences of services provided by lower layers and project core services to underlying communication protocols. Opening up new information sources, especially in field and control layers, the integration layer offers core services that are applicable in several manufacturing application scenarios. In fact, the combination of core services and gateway entities realize the required flexibility and interoperability. Aware of the real-time requirements in e.g. in production operation management, a service-oriented architecture is not applicable in all fields of manufacturing. Therefore, the integration layer is not considered a holistic approach. It rather seeks to identify appropriate areas to apply services in the manufacturing domain.

In the future we will conduct further research in the realm of service granularity, degree of coupling, and physical distribution within the integration layer. Since field devices will gain intelligence and applications will get more sophisticated, the question arises how close services may settle to lower and upper layers possibly replacing some gateway entities. We see sophistication of applications in near future and argue that our integration layer accelerates the development toward unified service-based applications in layer 3 and 4. We hope to extract additional findings by further implementing the integration layer and applying new use cases to it.

References

[1] IEC, "IEC 61158: Digital data communications for measurement and control Fieldbus for use in industrial control systems", 2003.

[2] IEC, "IEC 62264 Enterprise Control System Integration", 2003.

[3] M. Wollschlaeger and D. Hasler, "Uniform Identification and Maintenance Functions for PROFIBUS Devices as an Example for Web-based Information Systems in Automation (WFCS'2004)", in *5th IEEE Workshop on Factory Communication Systems, Wien*, 2004, pp. 385–388, Proceedings, IEEE 04TH8777, ISBN 0-7803-8734-1.

[4] OPC Foundation, "OPC Data Access Custom Interface Specification 3.0".

[5] OPC Foundation, "OPC Unified Architecture Specification Part 1: Concepts Version 1.00".

[6] C. Alexakos and et. al., "Workflow - Coordinated Integration of Enterprise / Industrial Systems based on a Semantic Service - Oriented Architecture", in *ETFA2005 10th IEEE International Conference on Emerging Technologies and Factory Automation, Catania (Italy)*, 2005.

[7] J. Gialelis and et. al., "Advanced Enterprise Process Modelling Utilizing Ontology Semantics", in *ETFA2005 10th IEEE International Conference on Emerging Technologies and Factory Automation, Catania (Italy)*, 2005.

[8] PROTEUS Project Homepage, http://www.proteus-iteaproject.com/.

[9] MIMOSA, "OSA-EAI V3.1 Product Release", http://www.mimosa.org/downloads/40/specifications/index.aspx/.

[10] SIRENA Project Homepage, http://www.sirena-itea.org.

[11] F. Jammes, A. Mensch, and H. Smit, "Service-Oriented Device Communications Using the Devices Profile for Web Services", in *3rd Workshop on Middleware for Pervasive and Ad Hoc Computing, MPAC 05, Grenoble (France)*, 2005.

[12] SOCRADES Project Homepage, http://www.socrades.eu, 2006.

[13] J. Peschke, A. Lüder, and H. Kühnle, "The PABADIS'PROMISE architecture - a new approach for flexible manufacturing systems", in *ETFA2005 10th IEEE International Conference on Emerging Technologies and Factory Automation, Catania (Italy)*, 2005.

[14] PROFIBUS International, "PROFINET Profile Guideline PROFINET and MES, Part 1: Maintenance Operations. Version 0.9", March 2006.

[15] Sun Microsystems Inc., "Java Business Integration (JBI) Specification", http://www.jcp.org/en/jsr/detail?id=208.

[16] Sun Microsystems Inc., "Open ESB Project Homepage", https://open-esb.dev.java.net/.

[17] D. Chapell, *Enterprise Service Bus*, O'Reilly, 2004.

[18] Red Hat, "JBossESB - Reliable SOA infrastructure", http://labs.jboss.com/jbossesb/.