Shop Floor Information Management and SOA

Konrad Pfadenhauer[1], Burkhard Kittl[1],
Schahram Dustdar[2], and David Levy[3]

[1] Vienna University of Technology, Institute for Production Engineering,
Karlsplatz 13, 1040 Vienna, Austria
{pfadenhauer, kittl}@mail.ift.tuwien.ac.at
[2] Vienna University of Technology, Information Systems Institute,
Distributed Systems Group, Argentinierstraße 8/184-1, 1040 Vienna, Austria
dustdar@infosys.tuwien.ac.at
[3] University of Sydney, School of Electrical and Information Engineering, Room 327,
Engineering Faculty Building J13, Sydney, NSW, Australia
dlevy@ee.usyd.edu.au

Abstract. Service Science is a new term for a new paradigm which aims at the solution of an obvious problem: How to make the increasing fusion of business and IT successful in a dynamically changing and risk adverse environment? This question has to be raised at different levels of abstraction, from macroeconomic viewpoints circulating around qualities of service societies to service oriented architectures of business applications. We worked in an interdisciplinary team consisting of industrial engineers and distributed system experts on the issue of business and IT alignment in a well-defined system, namely the shop-floor domain in discrete production industry. The result of this work is an ANSI/ISA 95 compliant model-driven methodology for manufacturing operations management. This methodology was evaluated by means of the realization of a SOA (Service Oriented Architecture) demo scenario for production operations management.

1 Intelligent Manufacturing Information Systems

Discrete manufacturing shop floor information and control flow management is still a challenging task due to the heterogeneity of data structures and information systems (automation components inclusively). The objective of vertical integration from high-level Enterprise Resource Planning (ERP) to the machine level is still unrivalled. Existing solutions led to static process logic coding within monolithic Manufacturing Execution System (MES) utilizing elaborate interfaces for rudimentary integration, lacking the needed flexibility and scalability. This proceeding is not sufficient regarding the requirements of today's dynamic production environments.

Internet based manufacturing, leveraging the latest technologies to achieve distributed information systems, provides new possibilities not only for static, data centric integration of the shop floor into an overall enterprise architecture, but also for full process integration of control and thus field level by means of SOA.

J. Eder, S. Dustdar et al. (Eds.): BPM 2006 Workshops, LNCS 4103, pp. 237 – 248, 2006.

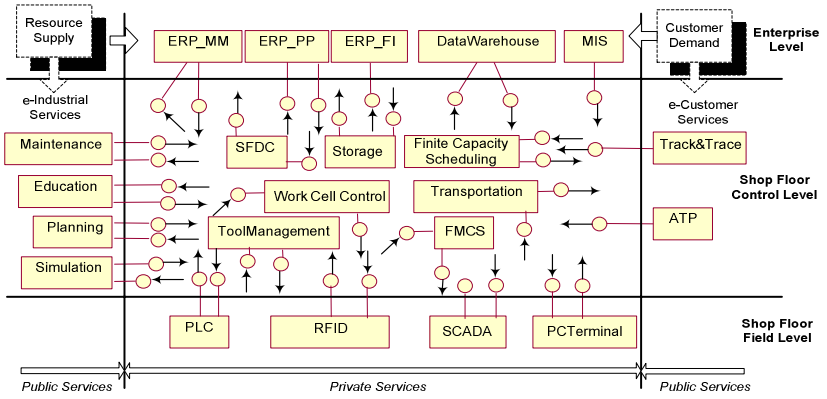

Fig. 1. Internal and external service providers at the shop floor control level

We assume that sub-system vendors at the control level (e.g. for tool management or for storage systems) will follow the trend towards service orientation already visible for ERP level modules. More control tasks get transferred to the field level PLCs (Programmable Logic Control) or open PC terminals, where the knowledge is concentrated and reaction times are the shortest. Hence Fig. 1 depicts potential service providers at different levels of the shop floor hierarchy, resulting in a complex distributed architecture which demands for system modelling and process life-cycle management. Public services into and out of the shop floor are next to the outlined vertical, mainly internal integration an issue too. Machine vendors are offering services like online maintenance, education support, planning and optimisation of production and logistics systems or other e-Industrial/tele-services. In our opinion this public, horizontal connectivity is in the short run not as promising as the vertical service integration due to higher coordination demands and security reasons. Recent developments at the machine level (e.g. Radio Frequency Identification) strengthen the call for an intuitive, model-based overview about service distribution and communication networks within the complete architecture.

The aim of this project was to investigate the potential of SOA for information and control flows in the shop floor domain, integrating applications as well as human workers as loosely coupled service providers. A modelling methodology was developed, which brings together system modelling at different levels of abstraction as well as process detailing and implementation at the execution level. Thus by means of existing standards concerning modelling (ANSI/ISA 95, UML (Unified Modeling Language)) as well as implementation technologies (Web Services, UDDI (Universal Description, Discovery and Integration)) Business and IT alignment was established.

The structure of this paper is as follows. We first discuss in Section 2 some manufacturing domain information architecture proposals. In Section 3 we present our model driven service architecture methodology. After that the implementation environment for a demo scenario is outlined in Section 4. Section 5 demonstrates the application of our *MDSA (Model Driven Service Architecture)* methodology by means of the demo scenario. With a conclusion and an outlook at the work to come we will finish this paper.

2 Manufacturing Domain Information Architecture Proposals

Enterprise Architecture (EA) research resulted in a number of elaborated architecture proposals with a more general scope (Zachman Framework, PERA (Purdue Enterprise Reference Architecture), GERAM (Generalized Enterprise Reference Architecture and Methodology)) as well as a manufacturing scope (CIMOSA (Computer Integrated Manufacturing Open System Architecture), GRAI Integrated Method). In our opinion these concepts introduce important ideas regarding modelling, modularization and abstraction levels. For instance the basic principles of MDA (Model Driven Architecture) or SOA can all be found in CIMOSA. Regarding modelling techniques applied as well as the assumptions made at the implementation level lack of standardization is the major problem of these architectures. We believe a feasible approach has to take the given techniques and technologies at the execution level into account and embed them into a broader architecture which supports Business and IT alignment. A joined initiative for manufacturing domain object and control flow standardization is ANSI/ISA 95 [2], [3], [4], a proposal derived from PERA. The limited scope, the use of UML and the focus on the higher abstraction levels as with the corresponding information flows makes this a very promising approach which we utilize and extend towards implementation level modelling. Proposals for BPM (Business Process Management) at the implementation level leveraging enterprise application integration, workflow or more recently process markup techniques (BPEL, BPML, XPDL) are promising, but concepts are missing how the integration into an overall platform independent enterprise architecture can be established. For a detailed discussion of state-of-the-art techniques and technologies concerning EA two EU initiatives delivered excellent publications [5], [6]: The aim of INTEROP (Interoperability Research for Networked Enterprise Applications and Software, launched 2004) is the conceptual as well as the technical integration of business by means of reference models. Contrary to our project the inter-enterprise system integration focus is dominant. Nevertheless, the chosen approach of MDA and SOA alignment, together with semantic annotations, shows some similarities to the approach presented in the following. But the INTEROP deliverables remain at a conceptual level, whereas in this work a domain specific real-world implementation proofs the quality of the methodology. Whereas INTEROP is the nucleus mainly of the university research community, ATHENA (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications, launched 2004) is an IT industry platform. Although useful but abstract reference models were available from the very beginning, very little relevant information was published how they can be implemented. Lippe et al. [7] demands for a 3-level modelling approach (Business, Technical and Executable Processes) and claim that a process abstraction concept is missing in existing architecture proposals. In their survey on modelling languages they claim that UML does not support business context, but in such a comparison the UML extension mechanism should be considered. All the more, as suitable UML profiles are provided for model driven SOA development (Berre [8]: UML Profile for PIM4SOA; Pondrelli [9]: UML Profiles for Services, Business Objects and Ontologies). In Pondrelli [10] it becomes clear that no new profiles are delivered, but existing proposals (e.g. IBM UML 2.0 Profile for Software Services) are incorporated in a rudimentary methodology.

Berre [8] presents the ATHENA Interoperability Framework. It would have been interesting to get more information about the ATHENA Service Oriented Interoperability Framework or the proposed MPCE Architecture (including Platform Independent Model for SOA (PIM4SOA) & Model Transformations) beyond the description in INTEROP D6.1 [11], but the content published so far is not sufficient for a detailed discussion. In addition, the focus on cross-organizational business processes with a strong emphasis on OMG Meta-Object Facility related model mapping increases the scope which is therefore much broader then the objectives of the single modelling language, intra-organisational approach presented here.

Recently, more emphasis on Service Oriented Analysis and Design (SOAD) can be observed. Arsanjani [12] rediscovers the three model abstraction dimension of reference architecture proposals like CIMOSA when he states that the process of service oriented modelling consists of three phases, namely identification, specification and realization. But he correctly postulates that it can no longer be an exclusively and thus unsuccessful top-down approach of domain decomposition, but a combination of top-down, bottom-up (existing asset analysis) and middle-out (goal-service modelling). For our methodology we adopted the hybrid SOAD modelling approach of Zimmermann et al. [13] who suggests a combination of Object Oriented Analysis and Design, BPM and EA techniques. It is the aim of this work to enrich and unify these fragments towards a comprehensive SOAD approach. Moreover, the methodology has to be validated by means of real-world standards, techniques and applications. Due to the weaknesses of existing solutions, we want to build up a SOA for the shop floor, optimising the trade-off between flexible interconnectivity and network infrastructure complexity. To overcome a situation of vertical, interrupted processes and partly unavailable, partly static accessible functionalities we introduce our concept of a MDSA for the shop floor a combined top-down/bottom-up methodology realized in a tool for user friendly model creation. On a conceptual base, the abstract MDA and SOA concepts are adopted for and enriched with concrete technologies and tools to implement a real-world framework for the shop floor domain.

3 Model Driven Service Architecture for the Shop Floor

Our methodological considerations started with the domain dimensions *Business*, *Architecture* and *Application*, each with its own modelling concept. SOAD has to bring those three together. End result should be a platform independent model, which has to be mapped to the actual and potential system assets. Hence it becomes a platform specific model, which will loose some of its *Business* readability as implementation details are added. In Fig. 2 we depict the resulting methodology specifically for the shop floor domain. The before mentioned domain dimensions were replaced by the classical hierarchy levels, namely Enterprise Level, Shop Floor Control Level and Shop Floor Field Level (Fig. 1). In the past the modelling concepts where utilized separately at the levels as shown in the figure.

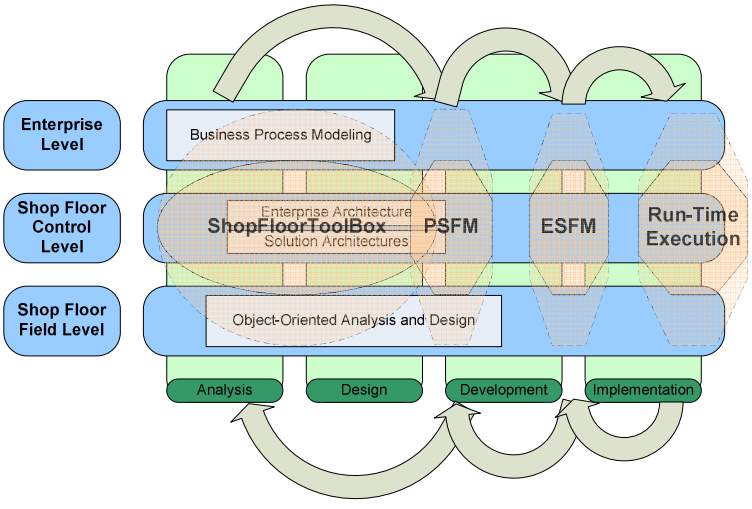

Fig. 2. Model Driven Service Architecture for the shop floor

First, fast and easy initial modelling of a given shop floor system has to be supported, focusing on functionality and connectivity of the system as a whole. We achieve this by a generic model collection called *Shop Floor Tool-Box (SFTB)*. The SFTB is an ANSI/ISA 95 compliant tool box which enables fast and standardized modelling of particular shop floor scenarios. The tool consists of an abstract service repository of basic and complex services (what dimension), concrete service providers (who dimension), binding mechanisms and data entities (with dimension). The *PSFM (Particular Shop Floor Model)* at the end of the Design phase can exist at two abstraction levels, platform/computer independent and platform specific. The latter constitutes the *ESFM (Executable Shop Floor Model)*. Whereas the PSFM will consider the actual system specification only roughly (coarse grain functionality distribution), the ESFM must be fully aligned with the assets either already existing or under construction. The PSFM has to support long term platform, infrastructure and service provider decisions through *as-is* and *to-be* comparisons. This high level model has to interact with the ESFM concerning process definition. The latter serves at a tactical level for the (re)design of service flow definitions which are semantically rich enough for executable code generation. Knowledge gained from the PSFM and ESFM should be fed back into the *Shop Floor Tool-Box*, which more and more becomes a valuable knowledge base.

4 MDSA Implementation – Assumptions, Technologies and Tools

We already mentioned the different approaches regarding SOAD. We strongly believe that only a combined approach can be successful, matching the given asset structure against high-level requirement business models. Thus we did a comparison of top-down and bottom-up approaches for model driven WS-composition, which led to the

result that satisfying technologies enabling stringent methodologies from business oriented system models down to executable service flow definitions hardly exist. Nevertheless, the approaches evolving around UML seemed most promising.

Therefore we implemented a combined *Specification and Representation* approach depicted in Fig 3, with a methodology based on UML 2.0 for high-level use-case and business scenario modelling (system model) resulting in platform independent service collaboration views (modified IBM UML 2.0 Profile for Software Services [14] as Model Representation Language). The *System Model*, which at a low level defines the composition specification, derives its syntax and semantic from a combined meta-model of an ANSI/ISA 95 Profile (leveraging the *Equipment/Functional Hierarchy Model* of Part I and the *Activity Models* of Part III) and the IBM Profile for Business Modeling [15]. Corresponding templates ease the model management. The repository of assets (service providers, realizing components) as well as other relevant information (data or binding types) are available in the low-level models and are incorporated in the fine-grained flow models (bottom-up). These flow models are the blueprint for the *Composition Model*, in our case BizTalk Orchestration Designer orchestrations. At this stage automated mapping between *Model Representation Language* (UML 2.0 Activity Diagram) and *Executable Composition Language* (XLANG/s) is not included. Other mapping requirements, which would gain importance if different modelling notations shall be integrated, are not relevant due to the limitation of modelling environments (Rational Software Modeler 6.0.1 with profile plug-ins and BizTalk Orchestration Designer 2004).

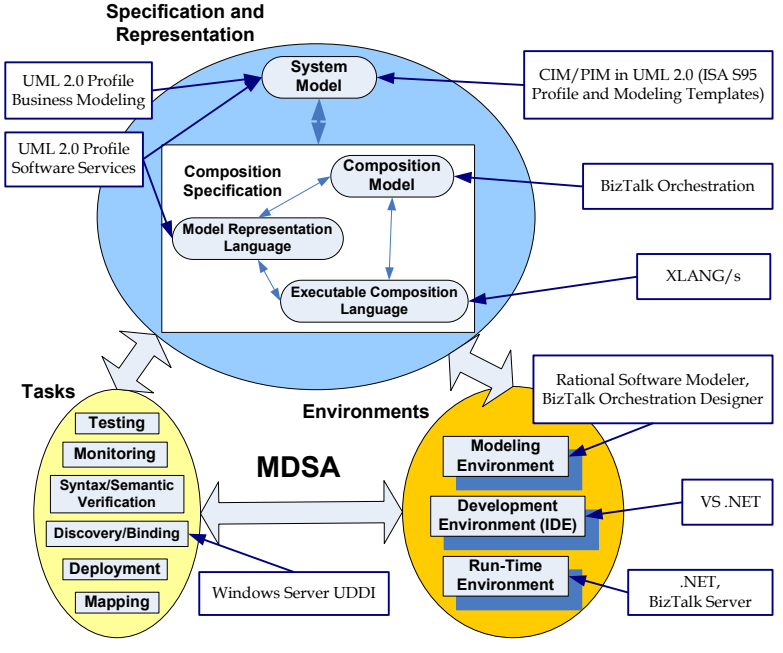

Fig. 3. MDSA implementation

We focused on discovery and binding, typically supported by a service broker, which is next to service provider and service requestor the third major role in the basic SOA paradigm. In the demo scenario this role is performed by the UDDI 1.0 compliant Windows Server 2003 UDDI. The Microsoft platform also prevails regarding the Run-Time Environment (.NET 1.1 and BizTalk Server 2004) as well as the Development Environment (Visual Studio .NET 2003).

5 MDSA Implementation by Means of a Demo Scenario

Within the modelling framework the complete ANSI/ISA 95 standards can be visualized and used as the guideline for domain modelling. Hence the methodology implementation follows ANSI/ISA 95 Part 1 concerning the general assumptions about hierarchies and functions. The generic *Functional Hierarchy Model* and the *Equipment Hierarchy Model* are the bases for functionality allocation and categorization. The implemented functionality in the demo scenario is part of the *Production Operations Management* grouping within the *Manufacturing Operations Management Model*.

ANSI/ISA 95 models are on the one hand the high-level framework for the SFTB, a collection of models and artefacts ready to be used for modelling projects. Thus the first task was to build an ANSI/ISA 95 UML Meta-Model in our modelling environment. On the other hand this Meta-Model constitutes the UML profile for ANSI/ISA 95, which is used throughout the modelling efforts. Mainly, the purpose of this profile is to keep the relationships to the business related ANSI/ISA 95 models and terminologies especially in low-level diagrams and models alive. Nevertheless, it is important to state that the SFTB is more than an ANSI/ISA 95 Meta-Model. It contains more information at different levels of abstraction and shall grow with every real-world modelling project, which means entities like service providers, data type definitions (OAGIS 9.0 Business Document Objects, OPC XML DA etc.) are continuously fed back into the SFTB.

5.1 From SFTB Constructs to Particular Shop Floor Models

For the remainder the Detailed Production Scheduling model shall be used to demonstrate the easy and highly integrated modelling methodology down to executable process specifications. The concept of modularization is important in the SFTB from the very beginning. That is why the use cases are separated from each other (Fig. 4). This gives the modeller the flexibility to first chose the constructs he needs and then integrate them, in the case of use case models through *include* and *extend* relationships or through the replacement of external actors and use cases.

Fig. 4 represents the highest level in the MDSA methodology. The whole Detailed Production Scheduling activity is represented as a *Business Use Case*, interacting with a *Business Actor*, the Production Schedule Provider. Typically, this role is realized by level 4 activities, often an ERP system releasing rough scheduled *Production Schedule*.

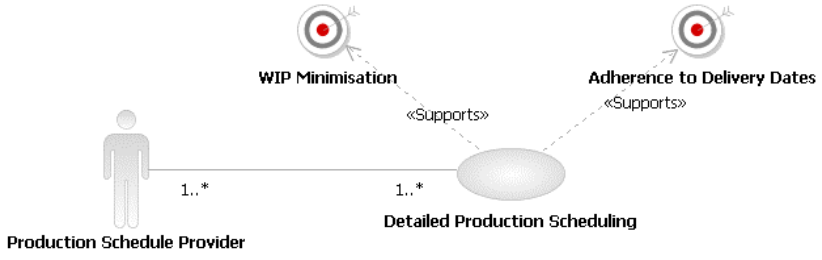

Fig. 4. Detailed Production Scheduling use case model

This use case supports two *Business Goals*, WIP Minimization and increased Adherence to Delivery Dates. The definition of goals is a key principle of process orientation, because the achievement of these objectives, refined by means of KPI, determines the effectiveness of the overall process. A control loop like the one presented in this work, has to translate the monitoring results of the operational level into figures representing the business goals. Business Goals are not included in the ANSI/ISA 95 standard and therefore give an example for constructs derived from the SFTB. *Business Collaborations* realize the use case. A centralized view provides the *Detailed Production Scheduling Realization Overview*, a diagram consisting of four diagrams according to the Rational Analysis Model template. However, alternative flow diagrams are optional and of course their number is unlimited. The template suggests using sequence diagrams for basic and alternative flows, but the use of activity diagrams or even non UML diagrams is possible. The first is a class diagram depicting all *Business Workers* and *Business Entities* for this particular realization. At least one dynamic behaviour view completes the basic overview. This is the second compulsory diagram in every Analysis Model.

5.2 From Particular to Executable Shop Floor Models

In an iterative process two modelling perspectives evolve, first the top-down derived PIM and secondly the bottom-up originating PSM. Thus the modelling project for a particular scenario can be seen as a central market place, were supply (the actual system configuration plus potential future functionality providers of the repository, part of the constructs of the SFTB) meets demand (the *to-be* system with its processes and goals defined by the business analysts). So far we have determined what *Production Operation Management Activities* we want to implement and modeled the high-level static and dynamic requirement models. Now it is time to have a look at which service providers are available and which roles they can play to realize the postulated system configuration. The service provider models can have two sources, either they are available as low-level constructs in the SFTB, or they are added to the project from the scratch. In either case, the modified IBM Profile for Software Services shall be used. Rational Software Modeler offers a template for this profile, which is used in an extended version for all modelling efforts at the software service level.

The service providers are grouped in a *Service View* perspective. We have already placed emphasis on the fact that the service provider construct is implementation independent. That is to say that the service providers, together with the interfaces respectively operations they realize, can be personnel as well as applications. To give an example: The *Detailed Production Scheduling* activity has an associated service provider role *SchedulingIFProvider* (Fig. 5), which provides for the service flow *scheduleNewOrders* the *IPreactorIF* interface (Fig. 6). This role is assigned to the scheduling software Preactor 9.2 interface extension. Another service provider is *SchedulingProvider*, a role assigned to the Preactor software user. All implementation details will be added in the subsequent step of *Component View* creation. The *Service View*, a component diagram, considers use dependencies as well as realization relationships. A class diagram and a composite structure diagram refine each single service provider component. For instance the *ProductionScheduleCheckProvider Service Provider* consists of two interfaces, stereotyped as *ServiceSpecifications*. Which operations they provide can be explored in the class diagram or in the general *Service View*. Each interface can be accessed at least trough two ports (*Services*), the general type is included in the port name (e.g. scheduleNewOrdersSOAP and scheduleNewOrdersFSO). The detailed type specification (e.g. SOAP-RPC, SOAP-DOC) is included in the documentation and has to be the same as the *Service Channel* binding attribute defined in the *Collaboration View*. This perspective provides a *Composition Overview* and a *Collaboration Overview* diagram. The first, depicted in Fig. 5, shows the relationships between the *Service Partitions*, which are collections of *Service Providers*. It is also shown what roles the partitions fulfil. In our case the partition is compliant with the activities of ANSI/ISA 95 Production Operations Management, thus we find a *Plant1:Scheduling* partition stereotyped *Detailed Production Scheduling*, which realizes the role of a *DetailedProductionScheduleProvider* for other partitions. Partition and UDDI are closely related.

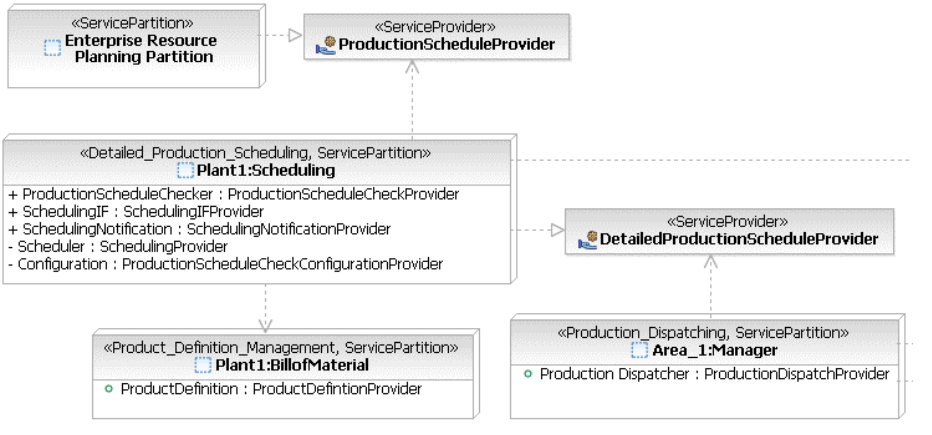

Fig. 5. Composition overview of the collaboration view perspective

We decided to map each *Service Partition* to an UDDI provider and every *Service* operation to an UDDI service due to the small number of operations. The UDDI categorization is ANSI/ISA 95 compliant as well, therefore the UDDI provider *Plant1:Scheduling* has the following categorizations assigned and provides all interfaces included in the partition: *Detailed Production Scheduling* (from ANSI/ISA 95 Activity Model categorization schema) and *Site* (from ANSI/ISA 95 Equipment Hierarchy categorization schema). *Service Collaborations* are assigned to ANSI/ISA 95 activities by means of stereotypes and refine the *Business Collaborations* described above. Each collaboration contains a composite structure diagram and at least one diagram for the behaviour view.

Fig. 6 depicts the static composite structure of the *scheduleNewOrders* collaboration. Here we see the participating roles and the provided interfaces (*Service Specifications*). The *Service Channel* stereotype contains the binding information. What we also see is that this collaboration depends on another *Service Collaboration*, namely *createPreactorImportBoM*. This collaboration gets bind by an internal BizTalk call, but offers a WS port (*Service*) as well. For three roles additional information (URL comment) is added. For *ProductDefinition* and *SchedulingIF* the URL points directly to the UDDI window, where detailed information about the actual implementation, e.g. the actual status categorization, is displayed. Those interfaces are statically bound, but *SchedulingNotification* can have a multiplicity greater zero, which means that within the collaboration a lookup for notification subscriptions in the UDDI takes place (*IInquiry* interface). The *DataDefinition* participant represents the XML schema definitions used in this collaboration, which are in this case deployed as .NET DLL at the BizTalk Server.

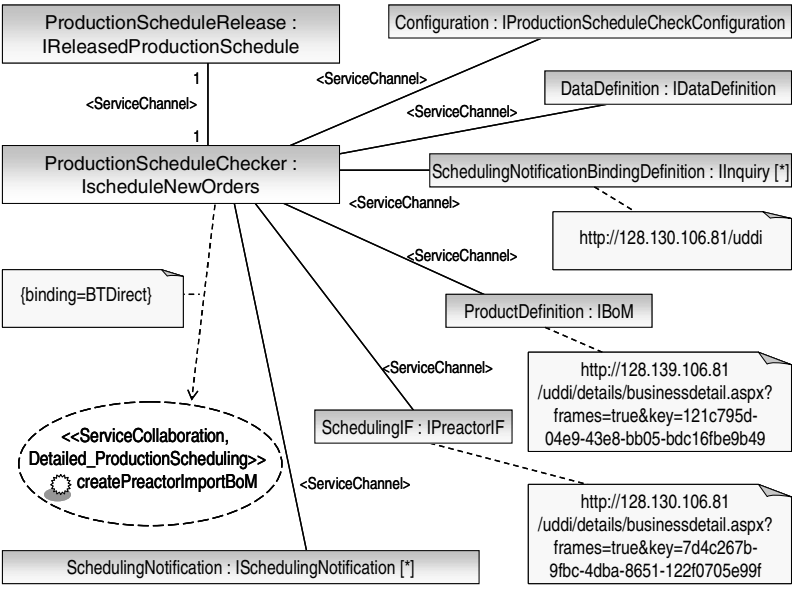

Fig. 6. *scheduleNewOrders* collaboration composite structure

The *Configuration* role provides access to the collaboration configuration, which has to be called as a separate BizTalk orchestration (including a business rule call for rules deployed at the BizTalk Business Rule Composer). The wrapping of the business rule call makes the rule platform independent, because the configuration helper orchestration can be published as a WS if necessary. Due to the shortage of space it is not possible to present the activity diagram containing the control flow which is the blueprint for the BizTalk Orchestration, although it remains platform independent in the sense that proprietary actions (e.g. *Transform* shape) are not included. What is added are again URL comments and business rule constraints. In addition, massage types are referenced by name for each object flow.

In the model organization message types and the assigned data types are collected in a *Message View* perspective. To sum it up, this activity model can be reused for different SOA implementation platforms. We have to mention the *Component View* perspective, where the realization of *Service Specifications* by means of components and classes are modelled. This OO abstraction level marks the end point for our methodology from business to enriched but still platform independent models. The mapping to the .NET environment and the BizTalk Orchestration Designer is the final step towards executable flow models.

6 Conclusion

The aim of this project was to investigate the potential of SOA in the shop floor domain and we proofed that this concept fulfills the requirements of state of the art intelligent manufacturing information systems. A SOA is flexible enough to realize decentralized control structures where appropriate and to integrate a broad range of service providers in a loosely coupled way. With the proposed MDSA methodology two gaps could be closed, resulting in business and IT alignment. First the gap to the implementation layer, which can be a very heterogeneous one in discrete manufacturing involving sophisticated web applications as well as manual processing tasks. The second gap is the one to the business layer, where business analysts define processes including goal and performance indicator setting. The outcome of this work is an ANSI/ISA 95 compliant model-driven methodology for manufacturing operations management. This methodology was evaluated by means of the realization of a SOA demo scenario for production operations management comprising of two dozens service providers, a central repository and user friendly terminal applications. It was possible to show that the proceeding is consistent enough to provide management capabilities throughout the whole system life-cycle. Moreover, the methodology is flexible enough to embed given shop floor scenarios and components smoothly into the framework with the help of predefined modelling constructs.

The successful participation of IT and domain specialists proofed the feasibility and user friendliness of the proposed methodology. At the implementation level it was interesting to see what restrictions a platform like MS BizTalk dictates in terms of system and not just single flow modelling. Especially the issue of nested flows made some proprietary patterns necessary. Next steps to come are some investigations regarding system dynamics. We would like to know how our approach performs in terms of control loops including flexible system adoption based on monitoring results.

Such a control loop concept must work at different levels of abstraction, providing every level with the right amount and granularity of information. Another focal point will be the interface between platform independent UML 2.0 activity models and the flow models of platform specific implementation environments. At the present stage this requires manual mapping.

References

1. IBM Research: Cover Story: Are we Ready for "SERVICE"?, Think Tank October 10th, 2005, Translated from Consultation magazine – ThinkTank Media group, accessed from http://researchweb.watson.ibm.com/ssme/20051010_services.shtml at March 15th, 2006
2. ANSI/ISA-95.00.01-2000 Enterprise-Control System Integration Part 1: Models and Terminology, ISA Organization, 2000
3. ANSI/ISA-95.00.02-2001 Enterprise-Control System Integration Part 2: Object Model Attributes, ISA Organization, 2001
4. ANSI/ISA-95.00.03-2005 Enterprise-Control System Integration Part 3: Activity Models of Manufacturing Operations Management, ISA Organization, 2005
5. ATHENA D.A1, Diez, A.B.G.(Document Owner): First Version of State of the Art in Enterprise Modelling Techniques and Technologies to Support Enterprise Interoperability, Deliverable D.A1.1.1, Version 1.0, July 2004
6. INTEROP D4.1: Scientific Integration Conceptual Model and its application in INTEROP, IST-508 011, Version 5.4, November 19th, 2004
7. Lippe S., Greiner U. and Barros A.: A Survey on State of the Art to Facilitate Modelling of Cross-Organisational Business Processes, SAP Research, 2005, accessed at http://www.athena-ip.org on March 24th, 2006
8. Berre A.-J.: Model Driven Interoperability – a standards based approach – and the ATHENA Interaoperability Framework, SINTEF, Presentation at eChallenges e-2005, Session Workshop 8°, October 20th, 2005
9. Pondrelli L. (2005a): A MDD Approach to the Development of Interoperable Service Oriented Architectures, Gruppo Formula, Presentation at eChallenges e-2005, Session Workshop 8a, 20th October 2005
10. Pondrelli L. (2005b): An MDD annotation methodology for Semantic Enhanced Service Oriented Architectures, accessed at http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-160/paper27.pdf on March 27th, 2006
11. INTEROP D6.1: Practices, principles and patterns for interoperability, Ed. David Chen, IST-508 011, Final Version 1.0, May 20th, 2005
12. Arsanjani A.: Service-oriented modeling and architecture, IBM developerworks, Nov 11th, 2004; accessed at: http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/ on December 12th, 2004
13. Zimmermann O., Krogdahl P. and Gee C.: Elements of Service-Oriented Analysis and Design, IBM developerworks, June 2nd, 2004; accessed at: http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/ on March 7th, 2005
14. Johnston S.: UML 2.0 Profile for Software Services, IBM developerworks, April 13th, 2005; accessed at http://www-128.ibm.com/developerworks/rational/library/05/419_soa/ on May 31st, 2005
15. Johnston S.: Rational UML Profile for business modeling, IBM developerworks June 30th, 2004; accessed at http://www-128.ibm.com/developerworks/rational/library/5167.html on May 31st, 2005