TU

TECHNISCHE UNIVERSITÄT WIEN

Dissertation

# Model Driven Service Architecture for the Shop Floor

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof.Dipl.-Ing.Dr.techn. Burkhard Kittl
E 311
Institut für Fertigungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Maschinenwesen und Betriebswissenschaften

von

Dipl.-Ing. Konrad Pfadenhauer
95 25 467
Lorenz Weiß Gasse 10 / 3, 1140 Wien

Wien, im Juni 2006

_____
eigenhändige Unterschrift

# Kurzfassung

Die Steuerung von Informations- und Kontrollflüssen in der diskreten Fertigung ist nach wie vor eine große Herausforderung. Dafür verantwortlich ist die Heterogenität der Datenstrukturen und der Informationssysteme einschließlich der Automatisierungskomponenten. Dies führte in der Vergangenheit zu einer statischen Codierung der Ablauflogik in monolithischen Applikationen unter Realisierung aufwändiger Schnittstellen. Diese Vorgehensweise wird den heutigen Anforderungen an eine dynamische Produktionslandschaft nicht mehr gerecht.

Internetbasiertes Produktionsmanagement versucht, durch den Einsatz von standardisierten Technologien verteilte Informationssysteme zu schaffen, die nicht nur eine statische, datenzentrierte Integration der Produktion in die gesamte Unternehmensarchitektur ermöglichen, sondern auch eine flexible, prozessorientierte Einbettung der Fertigungssteuerungsebene und somit auch der Feldebene in Form von serviceorientierten Architekturen.

Das Ziel dieser Arbeit ist es, das Potential von serviceorientierten Architekturen für Informations- und Steuerungsprozesse in diskreten Fertigungslandschaften zu untersuchen, wobei sowohl Informationssysteme und Automatisierungskomponenten als auch die beteiligten Mitarbeiter als lose gekoppelte Serviceanbieter integriert werden sollen.

Darauf aufbauend wird eine Modellierungsmethodik entwickelt, welche sowohl Systemmodellierung auf unterschiedlichen Abstraktionsebenen als auch Prozessdetaillierung und -implementierung auf der Ausführungsebene miteinander vereint. Somit wird mit Hilfe von bestehenden Standards sowohl die Modellierung (ANSI/ISA 95, UML) als auch die Implementierung (Web Services, UDDI) betreffend die Integration von Geschäftsprozessmodellen in die IT-Welt erreicht. Dazu bedarf es eines klar definierten Rahmens, der durch den Einsatz vorkonfigurierter Modellierungswerkzeuge (UML Profile und Vorlagen) möglichst einfach und anwenderfreundlich gestaltet sein muss.

Das Ergebnis dieser Arbeit ist eine ANSI/ISA 95 konforme Modellierungsmethodik für die Produktionssteuerung. Diese Methodik wurde an Hand der Realisierung einer serviceorientierten Demoarchitektur für die Auftragsabwicklung in der Produktion geprüft. Es konnte gezeigt werden, dass die Vorgehensweise konsistent genug ist, um die Systemarchitektur über den gesamten Lebenszyklus hinweg regeln zu können. Andererseits ist sie aber auch flexibel genug, um mit Hilfe vordefinierter Bausteine gegebene Szenarien und Komponenten einfach und mit genügender Genauigkeit in die Methodik einbetten zu können.

# Abstract

Discrete manufacturing shop floor information and control flow management is still a challenging task due to the heterogeneity of data structures and information systems inclusively automation components. This led in the past to static process logic coding within monolithic applications utilizing elaborate interfaces for rudimentary integration. 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 service oriented architectures (SOA).

The aim of this project is 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.

Consequently a modeling methodology was developed, which brings together system modeling at different levels of abstraction as well as process detailing and implementation at the execution level. Thus by means of existing standards concerning modeling (ANSI/ISA 95, UML) as well as implementation technologies (Web Services, UDDI) Business and IT alignment is established. Therefore, a well defined framework is necessary, which gains simplicity and usability through the utilization of preconfigured modeling tools (UML profiles and templates).

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 demo scenario for production operations management. 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 modeling constructs.

# Preface

## Motto

" Sie müssen auf alles selber draufkommen. Sie habe ja keine Aufgabe oder irgendwas. Aufgaben können nur Schüler haben und lehrerhörige Menschen. "

" You have to find out about everything by yourself. It is not like you have to complete an exercise or something. Only school-children and people who are submissively dependent on their teachers have exercises to complete. "

*Thomas Bernhard*

## List of Content

The structure of the thesis itself follows a top-down system engineering approach with different levels of abstraction.

We first discuss in Section 1 the motivation for this work emphasizing the growing importance of Service Science as an interdisciplinary research approach for complex and highly modular systems.

In Section 2 manufacturing enterprise systems and shop floor systems for discrete manufacturing are examined regarding system characteristics, integration approaches and modeling.

After that the research objectives are defined in Section 3.

Section 4 is dedicated to the three guiding principles and their specific occurrence in the context of this project, namely modularization (SOA), standardization (Web Service SOA and ANSI / ISA 95) and modeling (Model Driven Architecture, MDA).

In Section 5 the proposed model driven service architecture framework is presented, followed by the corresponding methodology description in Section 6 which includes the implementation environment outline for a demo scenario.

Section 7 then demonstrates the application of our MDSA methodology and framework by means of the production operations management demo scenario.

With a conclusion and an outlook at the work to come we will finish the thesis with Section 8.

Acknowledgement (to whom it may concern)

“ Dear Amigo ... Dear Partner--
Listen--I just wanted to say thanks. So ... thanks.
Thanks for all the presents.
Thanks for introducing me to the Chief.
Thanks for putting on the feedbag. Thanks for going all out--
Thanks for showing me your Swiss Army knife.
Oh and uh ... thanks for letting me autograph your cast. “

*Laurie Anderson, Big Science, Let X=X*

“ As you've been moving surely toward me
My soul has comforted and assured me
That in time my heart it will reward me
And that all will be revealed
So I've sat and I've watched an ice-age thaw
Are you the one that I've been waiting for? ”

*Nick Cave, (Are You) The One That I've Been Waiting For?*

# Content

# 1. Introduction

The work undertaken can be seen as an example for interdisciplinary research with the service concept as the embracing concept. The aim of a new service discipline is

> *… to create a service field to develop and carry out technical application so that help business, government and other organizations to improve current service and enter into new promising field. This field requires an understanding of how to create and deliver reusable assets so as to re-execute service more easily and deliver service more efficiently. This is the foundation of Service Science. This new discipline will combine the efforts in the computer science, operational research, industry engineering, business strategy, management science, social and cognitive science and legal science to foster the skills required by service-dominated economy.*
>
> *IBM Research (2006)*

This concept is the logical result of a paradox, which has become obvious in the last years after the collapse of the New Economy bubble. On the one hand there are technological innovations that are being advertised at an incredible pace, on the other hand there are risk adverse companies as well as humans who are increasingly reluctant to "cutting edge" promises. Thus the gap between not only business, but every day life dominated by all kind of services and IT broadens. In this environment service science proponents also can see the role for universities as providers of a combined Services Science, Management and Engineering education coming:

> *The method of business and technology combination requires high professional technologies. Obviously, it is impossible there is a researcher with doctor degree for every opportunity. Promotion closer integration of technologies and business requires new skills and combination of skills and new method of using these skills in the balanced way. These skills and application methods should be taught from university.*
>
> *IBM Research (2006)*

At the Institute for Production Engineering we realized this problem from a manufacturing domain perspective very early, although we did not call it Services Science. Our researchers at the Industrial Informatics Working Group always tried to invent business driven concepts and applications for the manufacturing domain and therefore required combined domain, IT- and system theory - knowledge. The work presented in this thesis must therefore also be seen in the context of this successful and industry- approved tradition.

A service is the representation of functionality, and functionality is a constituting characteristic of every system. Hence a system approach is needed. Within the system service provider and consumer must have a similar understanding of mutual expectations and responsibilities at all stages of the system life-cycle. In a dynamic, competitive environment, to close the gap between business requirements and operational execution of tasks, the following skills are essential:

- Ø Sound knowledge of generic system engineering principles
- Ø Knowledge of dominating influences (system processes customers) and key trends related to the system environment
- Ø System (a.k.a. Domain) structure and behavioural knowledge
- Ø Understanding of implementation technology

It seemed to be easier for our discipline to come to this conclusion regarding information affected systems, because we are aware of the obvious similarities between the quest for a business oriented software life-cycle approach and the concurrent or simultaneous engineering approach for manufacturing products. Here we see the concurrent development of market driven product specifications (business demand) and production facilities (assets). This approach is nowadays state of the art in manufacturing companies, and the curriculum of Industrial Engineering education can be seen as direct reaction of educational institutions to this development. An Industrial Engineer represents the link between management and workers, between visionary products for demanding markets and capable, reliable production facilities. We strongly believe the time has come to think about which role Service Engineering should play in the greater scheme.

Thus the Service Science approach seems to adopt the example of the materialized world for the world of information flows. Although the academic discussion is somewhat indifferent regarding scope and aims at the present (service industry Gross Domestic Product figures are mixed with SOA and Grid Computing), the discussion evolving addresses a core problem: What measures are needed to make the increasing fusion of business and IT successful, no matter in which domain? In this research project manufacturing is the business, and service orientation is the basic architectural principle. What is needed to fill the vacuum in between business requirements and implementation will be explained by means of a model based system approach. The methodology presented will proof our assumptions regarding knowledge and skills mentioned above. Call it Service Science or something else, but the IT solutions of tomorrow have to be highly interdisciplinary.

## 1.1. Motivation and Research Methodology

The aim of the work undertaken is to introduce the service concept in the context of information management to the shop floor environment. For the manufacturing domain with its heterogeneous information flows, heterogeneous information system landscape and complex control tasks, a number of methodologies and architectures evolved in the past. Most of them failed at the implementation level. There has always been a gap between theoretical architectural concepts and the technologies available for implementation. This does not mean that the concepts were inapt (see for instance the CIMOSA approach discussed in the following anticipated a lot of ideas which are now known as SOA and MDA), it means that technological restraints hampered the

actual application. A lack of standardization not only at a technical level (e.g. the variety of modeling techniques!) was one major obstacle, some kind of 'over-engineering' the other. The latter resulted in a weak acceptance rate of existing frameworks in the past. Hence one might think the actual implementation challenge is just about some enterprise application integration program work, resulting in some information flows across different application borders. Of course, at the implementation level with some new technologies like XML and Web Services as the unifying clue, this is a service approach as well.

In the author's understanding service orientation has to emerge out of a broader scope, it is not 'just a new bunch of techniques' for distributed systems. There has to be a framework, which tangles all abstraction levels of the enterprise's architecture. Hence a methodology has to guide all efforts toward service orientation. This methodology must be structured and presented in a way so that multiple people can work with it. Moreover, it has to have the clear aim to bring these stakeholders together and to give them a toolset which minimizes communicational frictions and efforts regarding business process management. Business analysts shall define the actual enterprise situation, its goals and its core processes. IT architects shall take this as their workspace and consider what kind of IT architecture can best support the enterprise in the given situation to achieve the goals. Or vice versa, business analysts are able to incorporate existing assets into the new architecture. And programmers should use the same blueprints to map the processes into the execution environments and adopt or code service providers according to these specifications.

The author sets the goals for this project as following:

- To develop a methodology which is feasible to guide business process implementation in a service oriented environment from high-level business models down to executable code and vice versa.
- To find a language which best serves the alignment of abstraction levels within the methodology.
- To examine the service concept for a shop floor application. Which activities in production management demand for what kind of service providers? How can legacy systems be integrated in service oriented architectures? What middleware techniques can be used?
- To answer the questions: What does service orientation mean for human beings in the shop floor? How can a loosely coupled architecture support the single worker in executing his tasks?
- During the writing of this thesis a lot of new questions arose, but the author tried to stay focused on the main topics mentioned above. Basically, the research work undertaken was following the scheme in Figure 1 (see below). Of course this thesis can not cover all aspects of the research project. Especially the literature review phase will not be completely incorporated in this text, only the central aspects which strongly influenced the project will be mentioned.

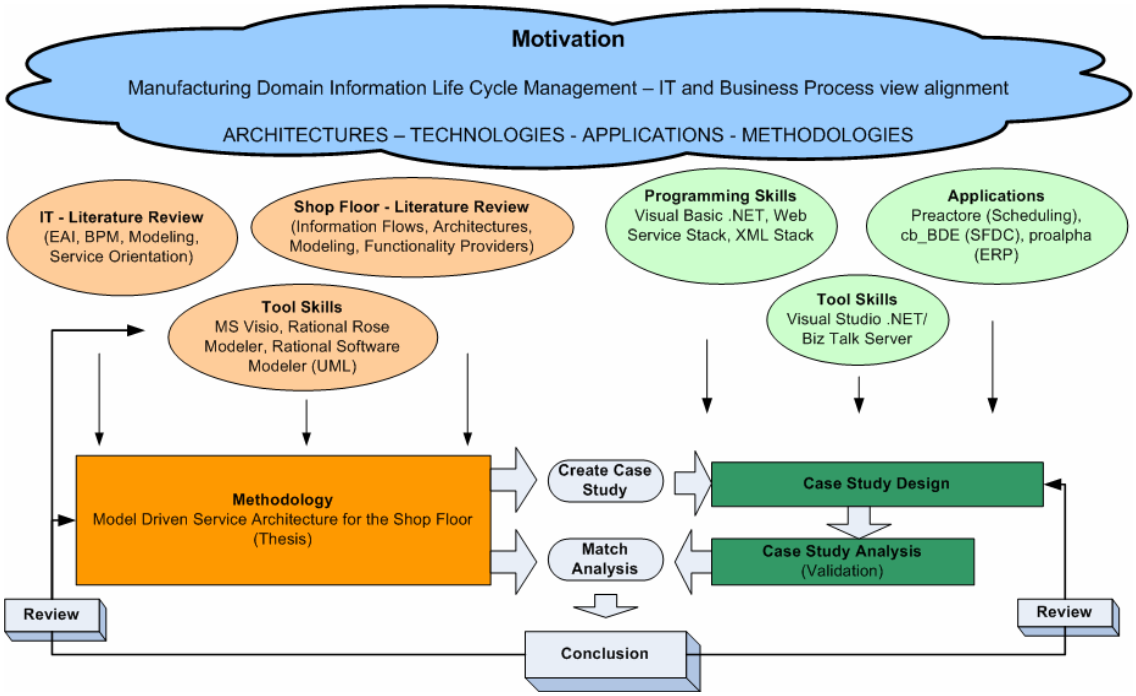


Figure 1. Motivation and research methodology

The system approach is the starting point. Without system engineering a task like the one outlined above can never be handled successfully. This conclusion becomes obvious when one realizes that there are many communities talking about similar issues. Enterprise Architecture, Business Process Management, Workflow Management, Enterprise Application Integration, Middleware etc. are all groups dealing with structures and processes IT has to enable. One of the big challenges of today is to bring together these concepts so that real world systems can be effectively supported.

It was thus necessary to examine the reasons why approaches to achieve this unification did not succeed. Some of the reasons for failure have been mentioned above but all of them were used in the end to develop a methodology that was more likely to succeed. Let us call these lessons learned the guiding principles for this project. A whole clause will outline what they are and how they are used. The system approach was not only essential for the methodology developed, but also to decompose the enterprise system so that the shop floor environment and its characteristics could be understood better. This analysis of today's shop floors in discrete manufacturing will make the motivation for the research more clear. The next step was to invent the methodology, which made some first decisions concerning techniques and tools necessary. Soon after that the creation of a first draft for the demo scenario started, and for the rest of the project the simultaneous engineering of methodology and case study proved to be very fruitful. In principle it was a permanent, iterative process of demo scenario design by means of the methodology and methodology improvement. What remained unchanged were the guiding principles, thus it can be said that these assumptions turned out to be correct.

# 2. System Approach

An open system can be best described as a collection of elements, which possess attributes that again are interconnected with each other and the environment and fulfil a specific purpose towards a common goal. Thus, systems are all around us; it just depends on the border setting, which again is determined by the objectives of the actor. Once the border is fixed, the activities of the system's elements together constitute the system functionality, which is defined for an open system as the ability to transform an input into an output.

Although historically developed during the 1940s to 1960s as a means to coordinate and control the development of complex systems, it is surprising how rarely architectural decisions are being made in a system engineering context today. In this respect Johnson (2003) outlines rightly that while it is possible to succeed with some programs some of the time without the social controls of systems management, it is impossible to consistently maintain high rates of success. Walford (1999) states that process automation requirements comprise process orientation and modeling and defines three supporting concepts for Business Process Management:

- systems engineering
- automation assets
- modeling

Explaining the importance of systems engineering, he states that

> *The most important message … is that a systems engineering approach to enterprise automation is the key to success. Without the discipline and unifying framework obtained with this approach, the maximum effectiveness of automation cannot come close to being achieved. That is especially true under the current conditions of rapid technology and business environment change.*
>
> *Walford (1999)*

Automation assets are the building blocks available to be integrated in a system. The traditional approach of system engineering does not emphasize the importance of reusable system assets and focuses more on a top-down approach. This has changed, and nowadays systems integration as a member of systems engineering gains importance. As mentioned in the introduction, the knowledge gained in respect to mechanical systems is highly adoptable for the information system domain. Prencipe (2003) gives a good manufacturing industry example for what he calls 'synchronic systems integration':

> *Systems engineering is a capability per se since it involves the identification of design compromise among subsystems, analysis of subsystems, and supervision of system testing (Sapolsky 1972). As found in the aircraft engine industry, after decomposing the product, engine manufacturers synchronize their work with that of suppliers and customers in order to assure the overall consistency of the system performance and to*

> *comply with the rules of the certification authorities. Synchronic systems integration should be seen as a two-way process. … Systems integration is a top-down process where engine makers model the engine, define the total system requirements, and break it down into components. Systems integration is also a bottom-up process where engine makers must be able to recompose what they have decomposed. Engine makers must be competent in both legs. Within a new engine development programme, engine manufacturers rely on state of the art component technologies and defined engine architecture.*
>
> *Prencipe (2003)*

It is easy to agree with Prencipe here and it will be shown that this two-way process is also a promising approach for distributed information system architectures. But what exactly are the requirements a two-way methodology for systems integration has to take into account? The standards for engineering management the US Department of Defence (DoD) define system element integration and verification like this:

> *The system elements shall be progressively integrated (bottom-up) into items that provide an end-use function. At each level, the resulting design requirements, physical configuration and physical interfaces shall be verified to ensure that the functional requirements are satisfied. The correlation of interfunctionally related elements shall be established and controlled. The techniques and procedural data for development, production, test/verification, deployment/installation, operation, support, training and disposal shall be determined, documented and implemented, as applicable. To provide a satisfactory solution set, each configuration item shall be evaluated to verify that it meets performance, functional and design requirements as well as user needs/requirements.*
>
> *DoD, cited from Johnson (2003)*

From the very beginning of systems engineering it was identified that many complex systemic problems are related to problems of communication, either miscommunication or a lack of communication. Therefore means to formalize and exchange conceptualizations were developed. The formal languages such as mathematics, symbol logic or drawings can be summarized with the term *model*. The concept of modeling for a better understanding of the enterprise system was introduced mainly by means of graphical notations, many of them at a rather poor formalization level. But while the demand for modeling is in theory accepted for system change or improvement, the reality is different, as the survey of Whitman and Huff (2001) shows: there is poor operational use of models and as a consequence existing business system models are rarely updated. A reason for this is that modeling is often seen as a unique task, which supports the process design once, not in a permanent process of change and improvement. Another identified problem is that different modeling techniques are applied to describe one and the same system. Walford's opinion regarding modeling techniques variety is that

> *Obtaining a good understanding of the structure and the operation of any enterprise, except for very small organisations, depends on the use of many types of modelling techniques. Unfortunately, the use of models in most enterprises is relatively infrequent. The models that are used tend to be somewhat informal and depend on the inherent knowledge of each individual involved for interpretation and utilization. Some recent attempts to introduce formal models such as the Unified Modeling Language (UML), which is used to model object classes, have tended to be very low level and specific to*

> *given topic areas. The lack of general formalisms worked reasonably well in highly structured, top-down organizations but presents problems when applied to the flat, more loosely structured organizations that are currently evolving. The management and automation needs of enterprises adapting these new organizational structures require that the business operations be defined and analysed in greater detail than before. Through the proper definition and application of models, the complexities and interactions of these new organizations can be better understood and managed. It is, therefore, necessary to define and utilize models in a structured and relatively rigorous sense. Without modeling, solutions to the complex problems inherent in the specification of enterprise automation can only be guessed.*
>
> *Walford (1999)*

What does the outlined system approach, incurred as the main principle for the work undertaken, really mean? Nothing more than the knowledge of the system as a whole, of the elements and the functionality each offers at the proper level of abstraction (together with the supporting infrastructure they constitute the architecture), no matter in what process they actually are involved in, because this is likely to change many times in the system life cycle. Fredrik gives a concise description of the importance of systems engineering in the context of future oriented system innovation when he states that

> *Innovation in systems technologies requires an understanding of specific component technologies, the functioning of the system as a whole, as well as the various components in the system design that need to be connected. These are the crucial aspects of systems integration …*
>
> *Fredrik (2003)*

The system theory is concerned with synthesis, analysis and methodology. It is the opinion of the author (see also Pfadenhauer and Kittl (2004b)) that system theory provides a wide range of proven and accepted concepts which are very useful in the context of domain architecture design, not at last in a Service Oriented Architecture (SOA). To accept the existence of general valid system theories makes it much easier to discover parallels between ostensibly different problem spaces and to transfer proven methodologies and rules into the own domain. Shi and Daniels (2003) provide an excellent example, how principles of manufacturing flexibility can be used in the context of eBusiness flexibility. They succeed because, in fact, they talk about system flexibility. In the following we want to demonstrate how results of the system theory fit into the world of Model Driven Architecture (MDA) and SOA. Modeling is the accepted key for this task. A model has to be an appropriate simplification of reality, and that was and still is the mightiest challenge: To define the level of abstraction and the granularity. With the concept of SOA it now seems possible to realize a MDA, a bi-directional and vital conjunction between the model of a system and the system architecture (Pfadenhauer and Kittl (2004c, 2005a, 2005b)).

## 2.1. System Manufacturing Enterprise

### 2.1.1. System Characteristics

Today's manufacturing companies are under the constant pressure of competitive and global markets as well as a highly dynamic system environment (Figure 2). Next to quality and costs the factor time is increasingly important. Time-to-market and time-to-delivery express the need for efficient and fast product engineering as well as order completion. Product development and order processes are just two examples for network enabled tasks, initiated by means of selling market events but involving all mentioned types of markets.

Resources are distributed between different locations, be it human knowledge or production capacity. The borders between internal processes and external processes blur due to close networking along the physical supply and virtual information networks. Like depicted below in Figure 2, the enterprise consists of a number of subsystems, which are highly interlinked, but not necessarily located at one site or acting with the same applications in similar organisational structures. And a subsystem like the production system can be decomposed further into production facilities scattered around the globe, one with a high automation level and the other relying on human workforce. But all the bits and pieces have to work together, putting increasing emphasis on system integration and therefore interface design (short lines in the figure). The business information applications like ERP (Enterprise Resource Planning) have realized the increasing demand for enterprise system integration not into a single, monolithic application but as a distributed system leveraging a flexible integration layer.

Hence, more and more information flows between all kind of markets and the enterprise system arise, revaluating the public interface layer. On top of this layer, which defines the interfaces and communication channels, the technological environment is placed, dictating the technology to be used. Technical issues like protocols or data exchange formats are embedded in the economical environment where business - needs are articulated and negotiated. This has to take place under the premises of the social, political and legal environment. An area which imposes increasing restrictions if one thinks of topics like corporate responsibility, Sarbanes-Oxley Act in US or Basel II in Europe. The ecological environment is surrounding everything, influencing the thinking and acting in the complete enterprise system as well as all markets. Sustainability can be a strategically means valued by the customers, a legal matter (e.g. Kyoto protocol) or a cost reduction factor (e.g. recycling).

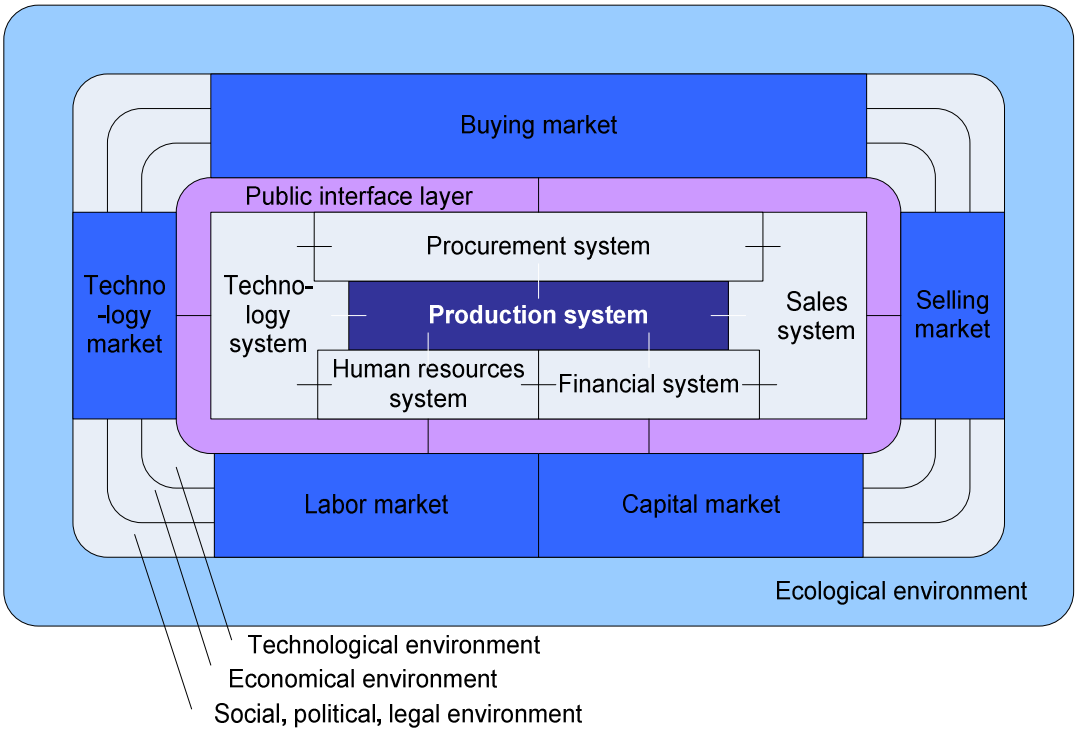


Figure 2. Manufacturing enterprise system view (adapted from Corsten (1996))

### 2.1.2. System Integration

System integration at the enterprise level is strongly driven by external needs. Business concepts like SCM (Supply Chain Management), Outsourcing, eMarketplaces, B2B or B2C etc. lead toward a horizontal integration (Figure 3) of companies as part of value networks. The figure is taken from the high level view of SCOR Version 6.1 (Supply-Chain Operation Reference model, issued by Supply Chain Council (2004)). This proposal is a good example for high level, inter-organizational frameworks which try to react to the increased connectivity between companies. But as in SCOR, the “Make” and “Enable Make” activities remain at an abstract level. The integration with the production environment remains unclear and thus is treated as a black box system. The process implementation level, be it physical or IT supported, is out of scope of SCOR. This increasing external interoperability also leads to the need for internal integration, especially at a data level. Data Warehouses or Business Intelligence are examples for concepts which shall assure a homogeneous data base with single points of access. A comprehensive state-of-the-art review regarding interoperability can be found in INTEROP D6.1 (2005), where good practices and solutions as well as principles/patterns for interoperability are provided. Especially the described UML based MACCIS architecture description framework implemented for the Norwegian Army has some relevance for this work regarding standard based modeling.

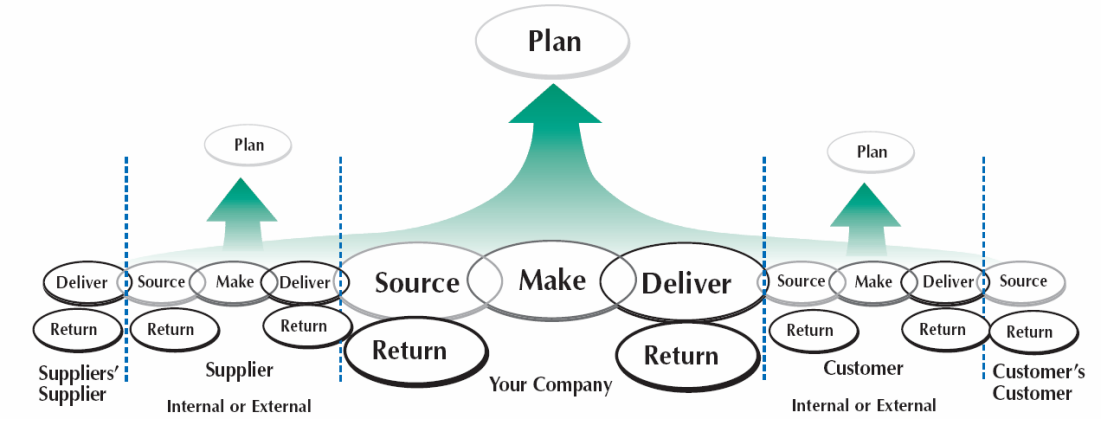


Figure 3. High-level horizontal enterprise integration (SCOR 6.1)

Beginning with EDI (Electronic Data Interchange), a number of languages evolved to enable inter-organizational data exchange, often enough with industry specific protocols and data formats. For a detailed discussion about those see ATHENA D.A1 (2004).

As substitute for many other standards and as an example for the technological environment in Figure 2 (s. pg 9) we shortly discuss RosettaNet, because it is one of the few concepts which tries to combine (rudimentary) process definitions with protocol layer definitions. RosettaNet combines the document or payload concept with short, choreographed point-to-point interactions called PIP (Partner Interface Process). This broader scope also incorporates time requirements. The RosettaNet Implementation Framework (RNIF 2.0) includes enabling specifications, mostly messaging, related to guide PIP execution (Kraemer and Yendluri 2002). This messaging approach supporting HTTP(S) over TCP/IP makes single PIP message exchange in theory adoptable for Web Services (RosettaNet 2003), but in practice the two concepts are too different (transactions, separation of payload and process) and hard to combine. Although RosettaNet Cluster 7: Manufacturing defines PIP specifications to manage manufacturing work orders or work in process (e.g. RosettaNet 2005), it definitely has to be seen as a B2B supply chain management standard.

## 2.1.3. System Modeling

### 2.1.3.1. Enterprise Architecture Modeling

An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprises. 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 modeling, modularization and abstraction levels. For instance the basic principles of MDA or SOA can all be found in CIMOSA, which is the EA approach that is most formally described and strongly committed to complete computer executability. What is interesting about this approach is that a complete system view including the integrating infrastructure is the goal. Computer Integrated Manufacturing Open System Architecture (CIMOSA) objective is the use of enterprise models for the monitoring and control of daily enterprise operations. To achieve this objective, CIMOSA developed two major parts: An executable enterprise model together with a two-dimensional derivation process (Figure 4) and an integrating infrastructure. CIMOSA enterprise model defines three model abstraction levels for the derivation of models (Requirements Definition to Implementation Description), as well as three abstraction levels for the customization of building blocks (from generic level to particular level building blocks, whereas the first two constitute the Reference Architecture). The four different views can be found in similar occurrences in many other proposals as well, for example in ARIS (Architektur integrierter Informationssysteme). The integration infrastructure uses the principles defined in DOAM (ISO/IEC DIS 10031-1 Distributed Office Application Model) for its IIS service definitions. DOAM can be seen as a SOA predecessor regarding service definition, encapsulation or request-response interaction patterns. At the time CIMOSA was developed, the integration infrastructures and technologies were not sophisticated enough to fully implement the approach. In this respect CIMOSA was years ahead, but it will be shown in this thesis that the time has come for a new “CIMOSA-light” approach which utilizes the available tools, standards and technologies. Regarding applied modeling techniques as well as assumptions made at the implementation level, the lack of standardization is the major problem of these architectures.

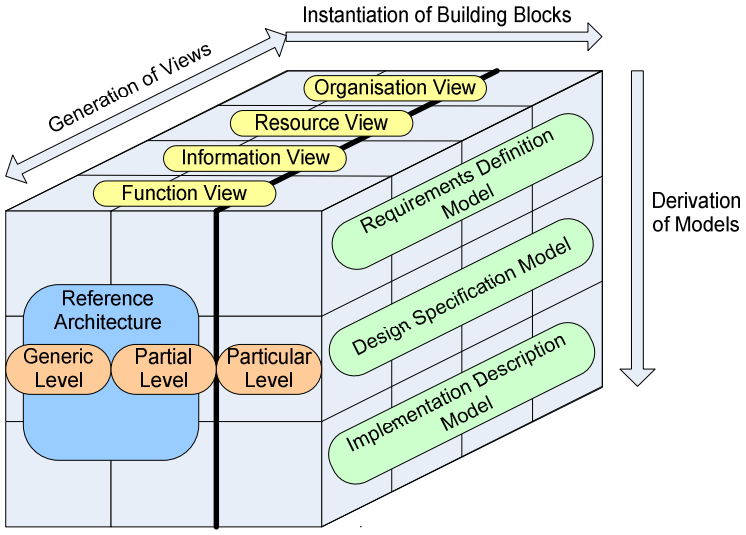


Figure 4. CIMOSA Reference Architecture

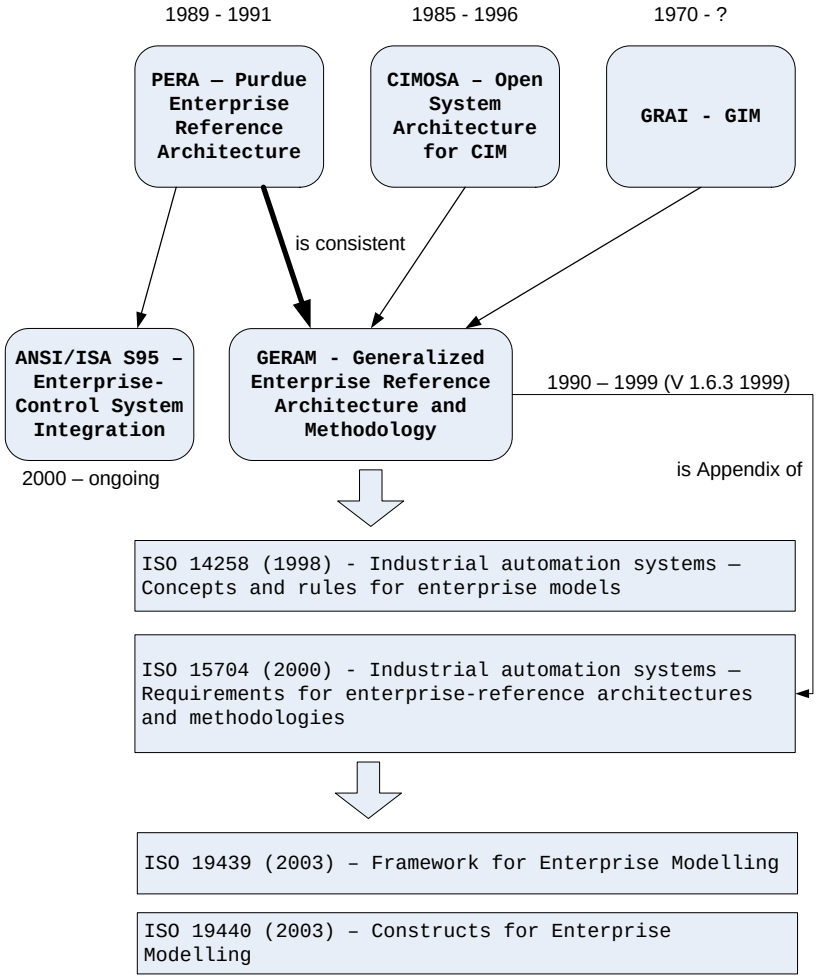


Figure 5. Major Enterprise Architecture Initiatives resulting in ANSI/ISA 95

The author is convinced that 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 (ANSI/ISA 2000, 2001 and 2005), a proposal derived from PERA (Figure 5). The limited scope, the use of UML (Unified Modeling Language) and the focus on the higher abstraction levels as with the corresponding information flows makes this a promising approach which the author utilizes and extends towards implementation level modeling. Proprietary EAI, workflow or, more recently, process markup techniques (BPEL, BPML, XPDL) are powerful when it comes to BPM at the implementation level, 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 it can be referred to two EU initiatives that delivered excellent publications. The first publication is the D.A1 of the ATHENA (2004) project, the second is a publication that summarizes the outcomes of the INTEROP project (INTEROP D4.1 (2004) especially for GRAI).

Generally it is interesting to see that government organisations, notably in the US, are heavily involved in the enterprise architecture and modeling research community. US military and government institutions defined the United States' Department of Defense's Architecture Framework (DoDAF). The framework is at the heart of the C4ISR (command, control, communications, computers, intelligence, surveillance and reconnaissance), which itself is used by the US military to support the planning, decision making, and execution of integrated battle scenarios. Moreover, the States of the United States, in response to the Clinger-Cohen Act, must demonstrate architectural planning for the budgeting process. The States have organized architectural bodies that govern and control IT architecture in compliance with the Clinger-Cohen Act. Much of their work utilizes the Zachman Framework. The Zachman Framework is a matrix of 36 cells covering the Who, What, Where, When, Why, and How questions of an enterprise. The enterprise is then split into six perspectives, starting at the highest level of business abstraction going all the way down to implementation. Such objects or descriptions of architectural representations are usually referred to as artefacts. The framework can contain global plans as well as technical details, lists and charts. Any appropriate approach, standard, role, method or technique may be placed in it. Although frequently looked at as a framework for building computer systems, the Zachman Framework is actually a classification scheme for descriptive representations of the enterprise as a whole, irrespective of its use of computers. The Zachman Framework has gained wide acceptance in the industry. Often used as part of a systems architecture or enterprise level technology review exercise it is popular within IT architecture departments but has little hold of either the developer or user communities.

#### 2.1.3.2. Business Process Modeling

Modeling of processes is a widespread accepted instrument for organisation (e.g. Business Process Reengineering (BPR)) or application (e.g. ERP system customization and implementation) design. In the latter case processes in a monolithic application are modeled, often enough with a tight coupling between the application and the modeling tool. The ERP software SAP and the ARIS toolset from IDS Scheer are a good but also rare example for that. How difficult it was and still is to merge the organisational process view with the application process view is being demonstrated by Hammer and Champy (1996) in their approved publication about BPR. Many of the success stories they cite (IBM Credit, Ford, Kodak) are based on increased IT coverage (EAI/portal solution for credit-check-process through generalists; central database for procurement process; simultaneous engineering and CAD/CAM). What they demand correctly is to avoid the trap of automating an inefficient process but to redesign it beforehand. In the 1990ies the productivity gains due to IT utilization were seen as a kind of miracle by business men, thus not only Hammer and Champy did not question what really happened at the execution level. At that time the process redesign together with first time IT adoption offered enough potential for improvement so that some inefficiency along the proceeding from good practice business processes down to the implementation level did not matter. Although it was

tried to cover all business processes in one ERP application, this carelessness resulted in the EAI problem as soon as it was realized that there will never be a single application or at least homogeneous platform solution for all business requirements.

Next to the obvious Business – IT gap, with system functionality increasing process durability became a problem. From a system theory perspective processes are partial behavior views within current system architectures. At a business level, processes are often decoupled from the overall system context and presented as the complete system behavior never to be changed. Thus, until recently, existing modeling approaches had a major disadvantage. For most of the applications a permanent semantic actualisation of the models was not necessary. Either these models were per se used only once (BPR), or they were stored for documentation purpose only (ERP reference models, ISO certification). Hammer and Champy introduced the so called Triangle, meaning that three alternative flows for every process should exist. Depending on the input parameters the most appropriate flow will be chosen. This selection mechanism is also known as business rule. Still, the processes per se are unchangeable. Due to the not foreseen or at least rather inflexible process redesign possibilities the demand for every-day operational usability could not be fulfilled. But with loosely coupled and distributed implementation architectures a direct and frictionless link between high level business process models and operational usability gains momentum. Nevertheless, at the enterprise level these limitations were already recognized and business process life-cycle management starts to influence modeling behaviour (Becker et al. 2003).

## 2.2. Subsystem Shop Floor

### 2.2.1. System Characteristics

The manufacturing type of interest for this project is discrete manufacturing. Discrete, continuous and batch manufacturing processes follow a classification generally used on specification of industry software products. This allows classifying manufacturing processes according to resource types and their relationship with time. For instance, the author examined the differences between batch and discrete manufacturing processes in terms of finite capacity scheduling (Pfadenhauer and Kittl (2003)). In the case of discrete manufacturing we operate with discrete time and finite/discrete resources. Product identification takes place on an item base, quantities are expressed by means of number of items and the base entity for product structures in production planning is the Bill of Material (BoM).

To manage the shop floor processes, one has to consider three flows, namely material, data and control flow (Leymann and Roller 2002). While process orientation leads to rather frictionless horizontal material flows, this is less true for control flows and least the case for information flows. Figure 6 (s. pg 15) tries to outline this situation.

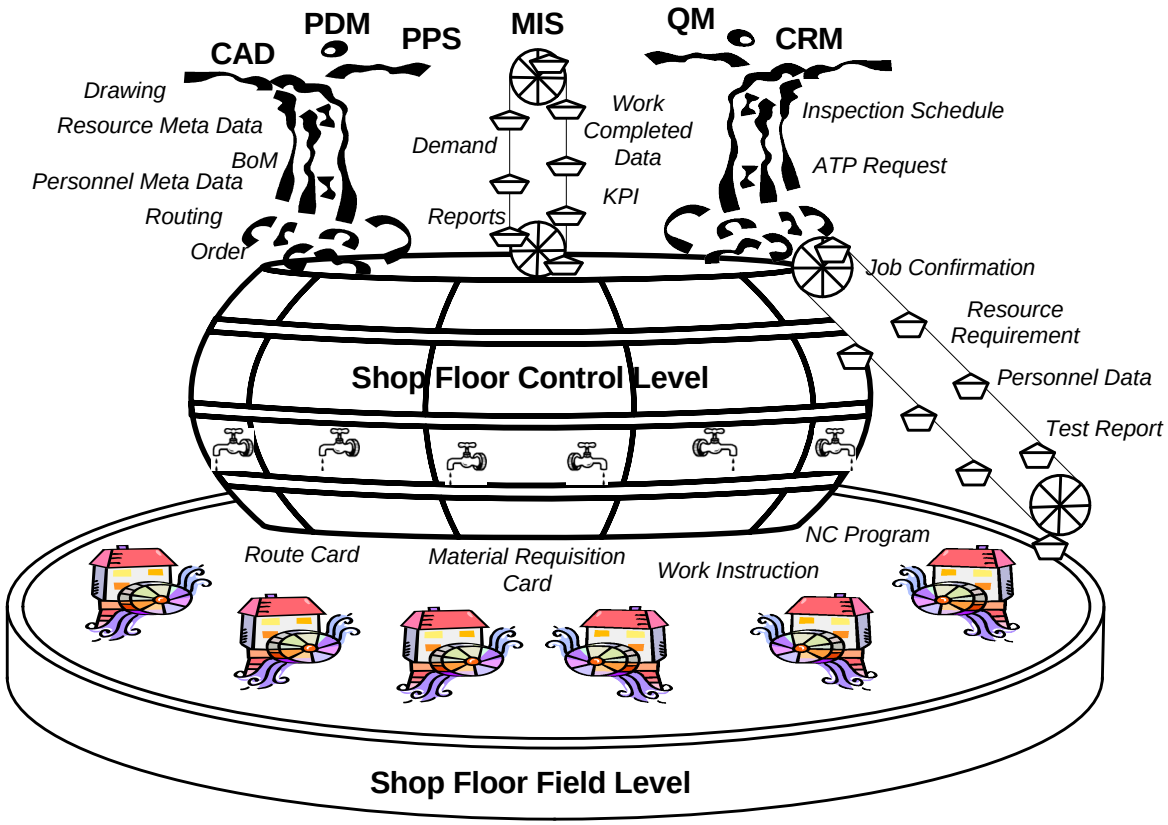


Figure 6. The shop floor system in discrete manufacturing

In this picture the material flows crossing the *Shop Floor Field Level* horizontally have been ignored. Waterfalls, taps and water wheels represent control devices. Control flows are most prominent between the hierarchy levels, of which at the highest mostly department specific applications reside. CAD (Computer Aided Design) is such a typical software, supporting R&D in product design. It is a matter of control structures if and when drawings are made available for the *Shop Floor Control Level*. Nevertheless, control relationships are present within the hierarchy levels as well. Enterprise-specific mechanisms exist, which define the control flows. Orders for example are released from the PPS (Production Planning System) if the material listed in the BoM is either completely in the storage or just for first operations. Or a foreman has to decide when to release work instructions at the *Shop Floor Control Level* depending on the actual resource utilization figures. Both, waterfall and tap stand for one-way flows, which means that it is always a more or less adequate information "shower" through well-defined channels which are controlled by sometimes simple, "water on / off" rules that sometimes depend on human experience. On the other hand the information flow back into the next higher level has to be controlled as well. *Shop Floor Control* and *Field Level* are both huge reservoirs of data, thus filter mechanisms are needed which extract portions the water wheel bins are able to handle. These reservoirs get filled from many different sources outside (waterfalls) and inside (mills)

the system. What makes the Shop Floor a tricky environment compared to, let us say, financial services is not the bare amount of data, because that is the same in e.g. clearing institutions. Two factors demand attention. First the data heterogeneity: the wooden barrel and the pool are full of different data formats with different life-spans. There are complex NC-Programs to handle, short route cards or unstructured data. Some of the data is relevant just for a small subsystem (e.g. transport data within a FMS (Flexible Manufacturing System)); other has to be provided to a number of different receivers (e.g. machine state data). One can find persistent as well as transient, real-time as well as batch run and XML (Extensible Markup Language) as well as hand written data. Taking into account this "Information Babylon", it is rather obvious that the information systems of the shop floor are heterogeneous too. Not only because of the data handled, but also because of system dynamics. A shop floor is a system of permanent change, be it NC Machines, personnel or other control systems. Thus the often changing hardware (although in Figure 6 all the mills look the same) demands for complex information flow solutions, which often enough can only be achieved by very specialized and customized systems.

This is true despite the fact that in the last decades more and more the IT became the driving force for efficiency improvement in the manufacturing enterprise.

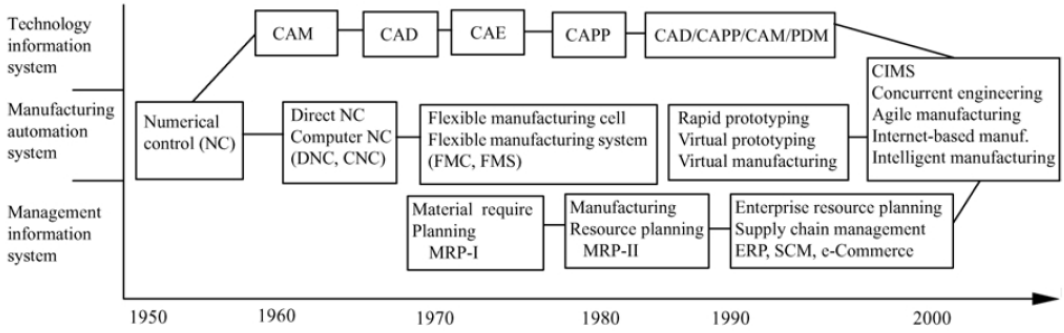


Figure 7. Evolution of IT-driven manufacturing concepts (Tian et al. 2002)

The most successful innovations were unambiguously concerned with autonomous solutions for partial problem spaces. CAD/CAM in combination with DNC and NC technologies solved the problem how to generate NC programs out of digital product data. But due to the collapse of CIM and the ghostlike "deserted fabric" the overall system focus was replaced by sub-system optimisation aims. Poor integration was the result, although certain approaches like the one presented by Kittl (1993) tried to keep an eye on the core (IDEF modeled) processes of the shop floor and not on arbitrary application borders. Only recently the discussion about overall information system integration was started again, unfortunately on two contrary abstraction levels.

1) First on a conceptual level, where "digital fabric" (Wagner and Blumenau 2003), "smart factory" (Westkämper and Jendoubi 2003), and "agile manufacturing" are examples for terms which shall give the changing manufacturing landscape a guiding framework. The major driving force behind the search for a new system integration

approach is the fact that the dynamism of the production system grows in terms of decreased life-cycles for both, products and production facilities. Shorter life-cycles of products on the markets demand shorter ramp-up phases, which again can only be achieved by means of simultaneous engineering of product and production. Shorter life-cycles of production facilities lead to faster changing functionality providers, but not necessarily to changing information and control flows. A modular approach with well-defined and standardized interfaces enables adoptability of the production system. In either case, system life-cycle management is crucial. An integrated, architecture model would also inhibit the growing functionality-overlaps through uncoordinated functionality extensions (e.g. an application was implemented although only 30% of the functionality are new; the 70% have to be supported redundantly) as well as data redundancy due to an inconsistent data management.

To give an example: with the data in ERP today no accurate accounting is possible. Cost objects get accounted only with indirect costs for machine-hours, tools, etc. Instead, if one thinks of the rapid growth of tooling costs in conjunction with the lack of cost transparency (Mumm 1997), direct cost information, which only the shop floor can deliver, is needed. Hence the demand for an architecture which allows to get the necessary information exactly from the objects which add the value (e.g. the tool-management, the single NC-machine, etc.), linking them to the accounting module via appropriate services. In Pfadenhauer and Kittl (2004a) a detailed description of this scenario based on UML models can be found. If we claim that not enough accurate data is available in the ERP system, exactly the opposite is true, too. One has to ask the question whether it is reasonable to centralize all the information in the ERP via EAI, fully aware of the fact that much of the shop floor data at the ERP level lacks the data quality and data completeness to provide a comprehensive picture of production performance. The new architecture should support information/data production and storage near the artefacts which create them; other constructs which need the data can call the appropriate services. Hence an Information Management System can be established very flexibly in this architecture. Such an IMS can be established at different levels of abstraction, either considering low level shop floor services with finer granularity or coarse grained services in a multi site scheduling scenario with the package shop floor as a whole, offering public services like order scheduling in an ATP (available to promise) process and hiding the nested low level flows.

2) These changes at a conceptual level described above are closely related to internet based, software driven, distributed environments at a technical level. Thus the discussion about system integration can also be found on the ground of network protocols and data objects. The reason therefore is the availability of internet based Field Area Networks (FAN), which promise a single network standard (TCP/IP) between the office application world and the automation devices. Although the potential of Ethernet is still under heavy discussion, it seems as if in the future FieldBus networks will prevail in critical real-time systems. Nevertheless, the future of distributed systems higher than the traditional sensor/actor networks will very likely be Ethernet dominated (Blecker 2003). Not alone the communication protocols within the manufacturing enterprise grow together, but also the processor hardware. Compared to the well-engineered, but insular DCS (Distributed Control Systems), PC

based systems offer more modularity and flexibility. They also increased their stability due to better OS. Hence PC based systems gain market shares on both levels, at operator terminal level close to the machine as well as SCADA (Supervisory Control And Data Acquisition) visualization level. One maturing technology will soon boost the demand for intelligent, bottom-up system integration: RFID (Radio Frequency Identification). Figure 8 presents an architecture proposal Siemens propagates for RFID and MES (Manufacturing Execution System) integration. RFID is the successor of barcodes for object identification. It depends on a wireless hardware infrastructure, which already demands for sensitive control tasks concerning access point installation bearing in mind net coverage or frequency overlapping. For an automated location detection and read / write activities on the tags in the context of process execution one has to have a model of the device architecture. Where are the reader / writer physically located, which interfaces exist with backend systems? Which services do they offer assumed active tags get used which request for information or want to transfer stored processing data? Siemens announced in October 2005 at the SPS / IPC / Drives in Nürnberg, Germany the integration of the new mobile UHF-reader Simatic RF610M with the MES component Simatic IT Production Modeler, now the central hub for RFID data transport into other applications (e.g. RFID read data transformation into a ERP receiving transaction).

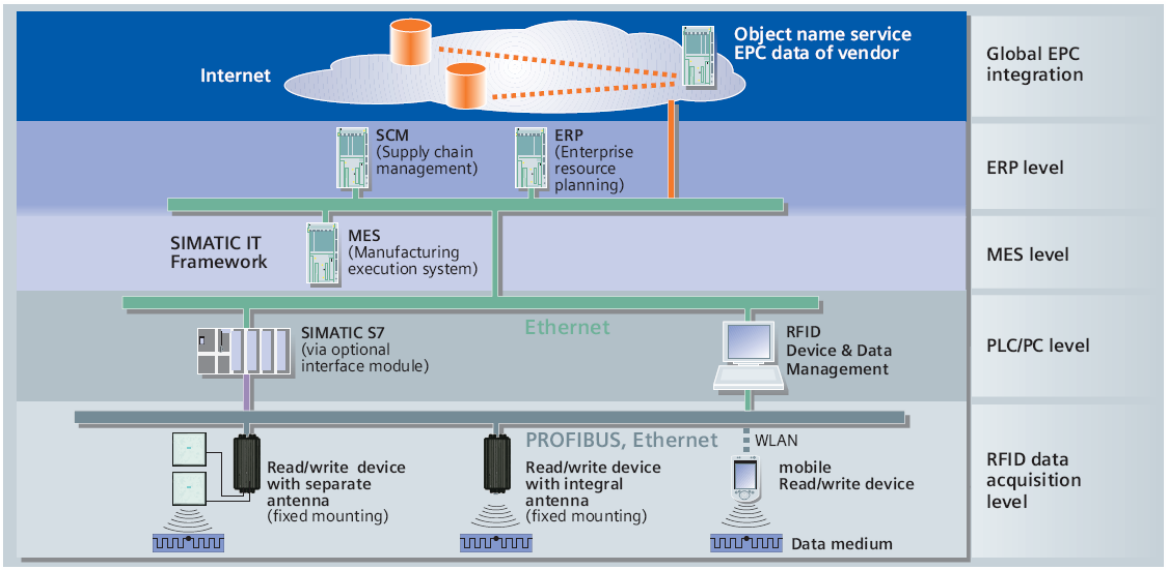


Figure 8. Siemens framework for RFID and MES integration (Siemens 2006)

The resulting standardization level regarding the technical infrastructure opens completely new possibilities for architectures which were missing in the past. Blecker (2003) underlines the importance of internet technologies and undertakes a comparison between approaches towards Internet-based production concepts (Figure 9). He correctly identifies the need for a strategic and methodical concept for modern production environments due to the increase of distributed services in production processes. In his opinion the depicted approaches do not establish a framework

sufficient in width and depth. In addition, he criticises the primary concentration on technical aspects. Nevertheless, Blecker agrees that a shift from hierarchical structures towards Intelligent Manufacturing Systems is noticeable and most prominent in agent based distributed systems. Intelligent service providers and service requestors behave like agents in a Multi Agent System (MAS), with own functionality and decision making capabilities. It is not hard to imagine that e.g. a tool with a RFID tag is integrated as a mobile agent into an existing MAS environment like PABADIS (plant automation based on distributed systems) (Pabadis.org 2004). Although the PABADIS framework claims to replace the centralized MES control structures by means of agents, it is obvious that the project members are part of the automation community and therefore the focus lies clearly on decentralized control at PLC and SCADA level. It is postulated that three out of eleven MES functionalities are commonly performed by SCADA systems, namely data collection and acquisition, process management and performance analysis. This may be true for the process industry, but even then the scope of the project is still limited. The title of deliverable 6.3, "Revolutionising Plant Automation" makes this more than clear (PABADIS 2001).

For Iung (2003) intelligent field components evolve toward MAS based e-Maintenance concepts embedded in a framework of CMM (integrated control, maintenance and technical management system). The formal maintenance platform modeling framework is based on the GERAM reference architecture, but in contrast to the work presented here it remains at the field component level for the process industry with focus on maintenance. Furthermore, it does not consider different modeling levels of abstraction.

| Concept / Criteria | E-Production / E-Manufacturing | Information-Based Manufacturing | e-Factory | WIM (IFF / PABADIS) | WIM (IJCIM) |
|---|---|---|---|---|---|
| Guiding Idea | • production in e-commerce<br>• optimization of the Supply Chain | • integration in Supply Chain Networks<br>• distributed, information-dependent production | • production as vertical element of the Supply Chain in e-business | • decentralized, agent-based automation as technical reply to turbulent environments | • interorganizational CAD/CAM combined with Internet Technologies |
| Aimed Conditions | • customer focus<br>• high quality<br>• low costs | • customer focus<br>• high velocity of (re)actions<br>• networked production<br>• synchronized demands | • process orientation<br>• low operation time<br>• production in an e-Supply Chain | • high flexibility<br>• adaptive<br>• reconfigurable subsystems | • flexible<br>• interoperable |
| Recommen-dations | • build-to-order<br>• e-technologies<br>• integration of customer & suppliers | • build-to-order<br>• Supply Chain coordination<br>• information sharing | • enterprise extension<br>• outsourcing<br>• inhousing<br>• cooperative manu-facturing operations | • distributed computing<br>• distributed automation | • high automation<br>• dislocated product development<br>• development of web-applications |
| Instruments | • e-procurement<br>• decentralized CAx<br>• decentralized production planning<br>• PDM/EDM | • web-EDI<br>• agent systems<br>• decentralized production planning<br>• Integration of IT and automation | • electronic machine control<br>• business information systems, e.g. ERP, CRM, SCP<br>• (e-)procurement | • agent systems<br>• embedded systems<br>• mobile code<br>• reduction of MES-System | • CAx or CAD/CAM<br>• quality function deployment<br>• enterprise appli-cation integration |

Figure 9. Comparison between production concepts based on internet technologies (Blecker 2003)

Blecker then introduces his own concept, Web-based Manufacturing and defines it as an

> *Internet Technology based Production Concept that is a (in theory) well-founded guiding-idea, based on empirical knowledge where appropriate, on the organization, planning, control and evolution of production systems. It aims at easily reconfigurable, high flexible production systems based on the comprehensive application of Internet Technologies on the shop floor.*
>
> *Blecker (2003)*

After emphasising the potential regarding interoperability and connectivity ("plug-and-produce") from individual sensors to office applications, he focuses on the consequences for as well as the coordination between management and operation subsystems. Although we are agreeable to his assumptions concerning the implications of internet technologies (Figure 10), the Web-based Manufacturing concept remains unexplained. Nevertheless, for the development of the methodology presented in this work, the potential of internet technologies not only highlighted by Blecker is essential.

| Machines & Facilities | Information Systems | Materials Flow System | Employees & Work Places |
|---|---|---|---|
| • integration of information systems<br>• standardized interfaces<br>• accelerated machine condition diagnostics<br>• high factor mobility<br>• low factor specificity<br>• parameterization and configuration in IP-based networks<br>• high flexibility | • integration of disjunctive systems<br>• compatibility to office systems<br>• high system transparency<br>• high information transparency<br>• low context incommensurability | • integration with machines and plants<br>• redesign | • diffusion of high-tech workplaces<br>• high information availability, even in decentralized work places<br>• multi-media based equipment<br>• changes in operating interfaces |

Figure 10. Potential Modifications of subsystems derived from Internet Technology (Blecker 2003)

This increased internet technology focus is visible in the *Machines & Facilities* sector, where the Ethernet makes its way down to the machine level, interfacing PLC and control devices. The International Electrotechnical Commission (IEC) reacts to this development and proposes an extension to the IEC 61131 standard for PLCs (Programmable Logic Control) in centralized architectures through the IEC 61499 standard for distributed control systems. Event interfaces and compliance profiles for the network and data exchange layer extend the basic paradigm of function-blocks. One of the major aims of this effort is to ease the modeling of automation processes (Favre-Bulle 2004).

High information availability for *Employees & Work Places* is a very promising field for internet based, distributed *Information Systems*. The high information transparency will result in changes in operating interfaces. The author strongly believes that with distributed system architectures the information retrieval process will become more pull-oriented, which means that the employees can design their own information cockpits flexibly and highly customized. Moreover, the employee should be empowered to easily include the information blocks actually needed. Again the ERP level with the new portal / portlets GUI generation provides a good example of what is possible, although most of these applications still reside on homogeneous platforms.

### 2.2.2. System Integration

Often enough the production sub-systems work stand-alone or the high integration effort allows only for minimal, vertical interfacing. Either the vertical integration gets achieved by means of direct ERP and sub-system interfacing, which increases the ERP complexity and results in huge amounts of row data at a level where proper interpreted one would be needed. Some sort of improvised data filter and transformation is therefore unavoidable. Moreover, crucial information is missing in this scenario. We discussed the lack of cost data quality (indirect costs-domination instead of direct ones is responsible for the well known cross-subsidisation between products with high and low profitability), which is a direct result of insufficient information integration. This enlarged ERP architecture stresses the functionality and scalability of the ERP system.

Or the integration effort of shop floor functionality results in a monolithic control centralization at the top of the shop floor hierarchy, which establishes the interface for higher systems like PPS or MM (Material Management). Typical examples are MES (Manufacturing Execution System, Figure 11).

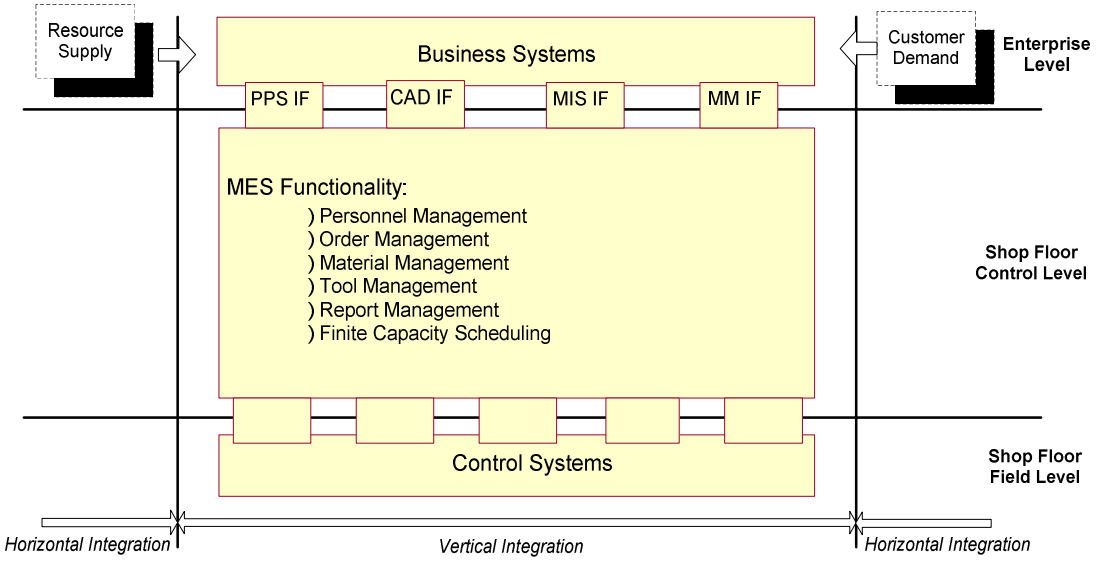


Figure 11. Centralized and monolithic MES integration architecture

In principle, MES objectives are to provide mission critical information about production activities across the enterprise and supply chain via bi-directional communications. Therefore, they aim to improve the return on operational assets as well as on-time delivery, inventory turns, gross margin, and cash flow performance. More and more regulatory compliance or quality programs become the main force behind increased MES interest. Figure 12 depicts the scope of a typical MES system. Although many existing MES solutions offer the functionality in a modularized form, we still call them monolithic, because the integration of third party products is not easy to manage once the modules are implemented. Siemens, for instance, with its Simatic IT product suite claims flexible process definition by means of a graphical modeler tool, but this is true only for Microsoft platform specific components and COM bindings (Di Salvo et al. 2003).

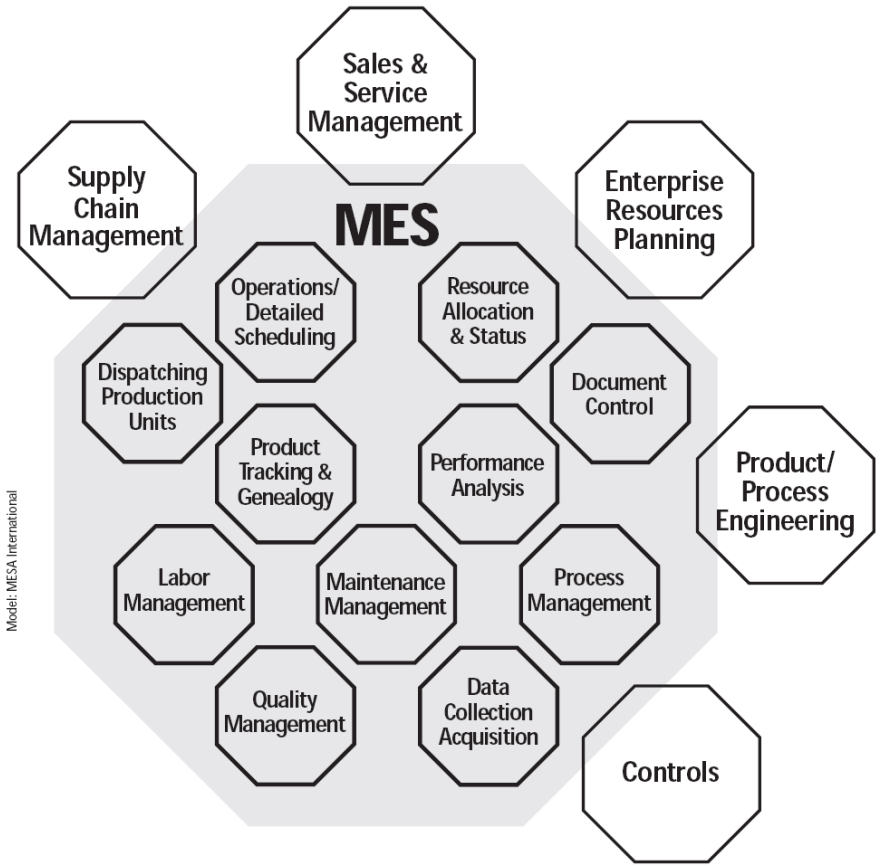


Figure 12. MES functionality defined by the MESA consortium

Bauer claims that MES suffer from static structures and high costs (Bauer 1995). In PABADIS (2001) the limitations of MES systems are seen mainly from a scheduling perspective, claiming that the growing product variety and complexity leads to an increased complexity for resource allocation and therefore scheduling algorithms. Second, the flexibility of MES is not sufficient when it comes to changes in the product mix. And last the problem of robustness and failure risk is mentioned. So far, data centric integration efforts prevailed, resulting in an extensive data-push into the ERP

systems. ERP systems typically offer API (Application Programming Interface) for these purposes. If this is not the case, often enough "innovative" solutions can be found enabling a direct ERP database table access. But MES systems mostly offer standardized adapters for the most important ERP packages. Nevertheless, almost every integration project is unique, because in this area no standardization was enforced and proprietary interfaces and data formats prevailed. And the number of proprietary MES products is unmanageable, as the yearly MES market survey of Frauenhofer Institute for Production Engineering and Automation and Trovarit AG (Wiendahl et al. 2004) proves. The products not only differ enormously regarding the functionality, but also the technology they implement. Many of them still do not provide Web Clients or Web Service interfaces, even XML support can not be taken for sure.

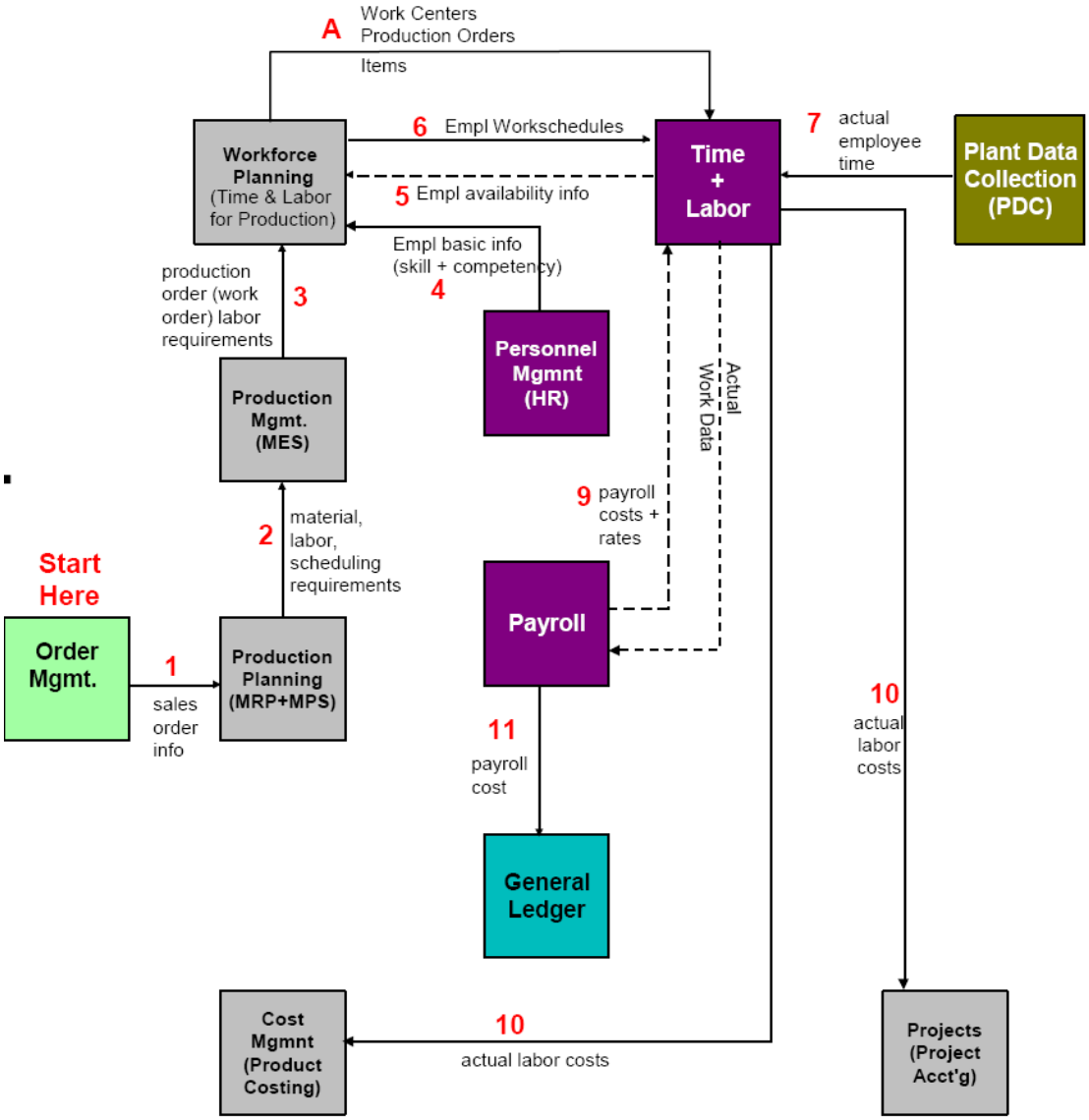


Figure 13. Open Applications Group manufacturing scenario (Connelly 2005)

Recently some institutions evolved which try to tackle the issue of standardization, like MESA (Manufacturing Enterprise Solutions Association) or ISA (Instrumentation, Systems and Automation Society, see clause 2.1.3) in the field of manufacturing operation functionality. The different standardization groups also try to merge their efforts toward a harmonized set of information standards in the industrial field. The Open O&M, occupied with operation and maintenance information exchange, is a joint initiative of Open Applications Group (OAGIS standard), ISA (S 95 standard), World Batch Forum, MIMOSA (Machinery Information Management Open System Alliance) (MIMOSA standard) and Open Process Foundation (OPC standard). Because there are many overlaps, the goal is to harmonize the key standards of the participating organisations.

Connelly (2005) describes in his work some scenarios (Figure 14) and processes (Figure 13) which demonstrate the scope of the joined standardization efforts and the areas to which each organization can contribute, but so far nothing else except such high level models has been published. Hence no complete framework including standardized modeling methodologies or patterns is available. Nevertheless, the interfaces and complex interactions depicted in those two figures underline the importance of modeling as a means of system management.

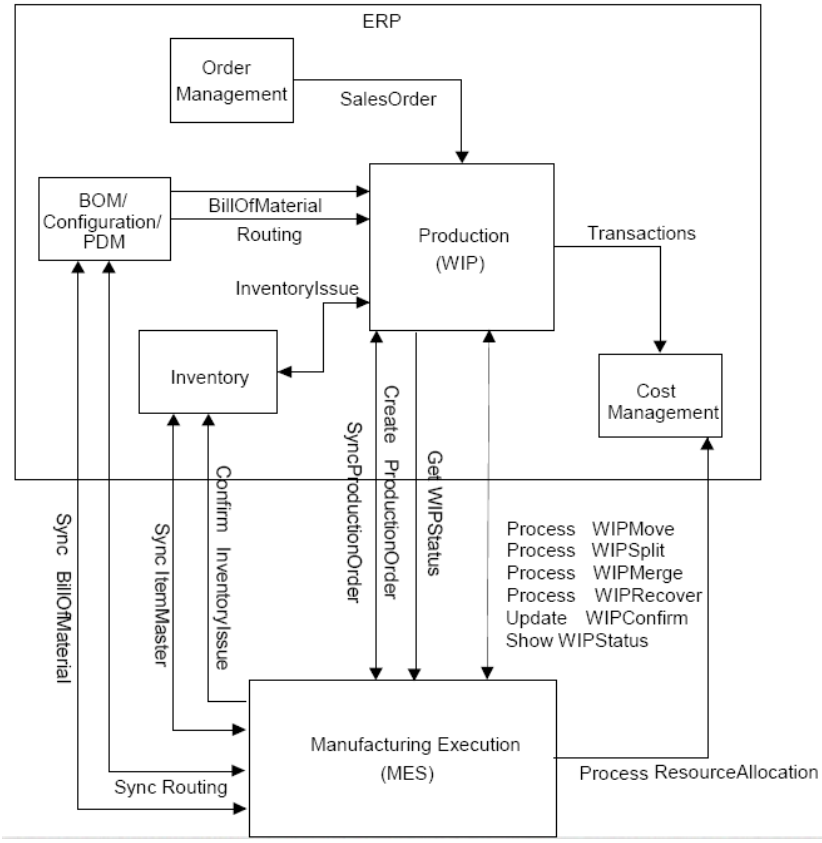


Figure 14. Open Applications Group “Production to MES” scenario (Connelly 2005)

So far the importance of systems integration between the shop floor and external systems regarding information exchange was limited. Most of the integration was achieved at an enterprise level (Figure 11). This "business application intermediary approach" is valid for most of the IT supported integration efforts.

### 2.2.3. System Modeling

The distinction between Business Process Modeling (BPM) and Information System Modeling (ISM) is especially present in the shop floor. On the one hand automated sub-systems like Manufacturing Cells are modelled with ISM in the context of control system development, on the other hand BPM seldom gets in touch with Tool Management and magazine handling of Manufacturing Cells, i.e. the abstraction level is rather high. Hence an alignment of both design domains is necessary. The CIMOSA approach (ESPRIT Consortium AMICE 1993) is in our opinion a good orientation point for this objective.

More physical oriented modeling takes place in the context of production facilities and logistic systems design in terms of product/process engineering. The so called "digital" or "virtual factories" consist of a complete digitised physical model of the factory, CAD/geometry/simulation data of systems like machines or transport facilities merge with product data, thus through data and process integration a shift from simultaneous to cooperative engineering is recognized (Sihn and Graupner 2003). Due to increased production environment changes the next step is towards the "smart factory" (Westkämper and Jendoubi 2003), which supports optimisation throughout the life cycle of the manufacturing systems and the depending processes (Wagner and Blumenau 2003). Again only for the system build-time in the design phase modeling methodologies for rather encapsulated shop floor sub domains, like FMS (Flexible Manufacturing System) were developed. But due to platform homogeneity of such islands of automation the demand for integration support by means of models was weak (e.g. ProduCASE tool from ABB described in Fröhlich et al. 2002).

The same is true for model based monitoring and control tools for automated systems (SCADA) in the field level, which allow system design within the vendor's platform and within homogeneous networks (technology or vendor spanning approaches rarely exist). New SCADA versions are now being designed to handle devices and even entire systems as full entities (classes) that encapsulate all their specific attributes and functionality and use TCP(UDP)/IP as protocol.

To sum it up, system life-cycle modeling becomes more and more important, especially in the shop floor domain, to cope with increased system-variety and connectivity resulting in high flexibility expectations for both, material and information processes.

#### 2.2.3.1. OO Shop Floor Modeling

The Unified Modeling Language UML is an ISM technique (Giaglis 2001), but well suited (Dumas & ter Hofstede 2001; Fröhlich et al. 2002) for ISM / BPM alignment (through variable extensions like profiles). Aguilar-Savén (2003) introduces in her business process modeling review a classification framework which displays UML on the top concerning variance of model purpose and model change permissiveness compared with other techniques. Only workflow techniques and GRAI-GIM perform better in her comparison. The problem with workflow techniques is, that a number of proprietary proposals exist which lack a certain degree of standardization to gain widespread acceptance. The GRAI-GIM approach strongly influenced the GERAM approach (Figure 5), but although it was an interesting approach invented for manufacturing processes, it did not succeed due to a low formalization level. Furthermore, it lacked vendor support. Proven UML engineering methodologies are available, all of them (e.g. in Ng 2002; Dietzsch 2002; Hitz and Kappel 2003; Mielke 2002) relate to more or less three views: the functional, the structural, and the behavioral. In Mielke (2002) the problem domain for a single application development project is well known and the use of cases (functional view) is therefore less important. Instead, an organization model is introduced as an extension. In Dietzsch (2002) the reasons why the UML Business Modeling Profile gets combined with parts of the PROMET BPR tool are mainly the lack of concepts for goals, critical success factors and index values of processes. Dietzsch describes that this method integration leads to various problems in practice, which is why we choose pure UML with the foreseen extensions in the Meta-Model.

Also much research has been undertaken concerning UML usability in the Industrial / Manufacturing Environment. Bruccoleri et al. (2003) give an example of how UML can be used as a system design methodology for Flexible Manufacturing Control Systems. While the methodology used by them is interesting within the scope of our considerations, there are still problems with this approach: clearly, a given static structure of a Manufacturing Cell gets modelled and the flexibility mentioned affects only changing attribute values and not changing operation distribution between the objects, which would be very important for the shop floor, where single operations / services can be offered by changing objects / components. How to handle the problem of changing service distribution, and thus the independence of services and components in the analysis phase, is demonstrated in the demo-scenario.

The point is that these papers show that UML together with its extension mechanisms offers sufficient freedom to cope with the shop floor system complexity.

#### 2.2.3.2. ANSI/ISA 95 framework

The work for ISA S95 (widespread abbreviation for the bunch of standard parts mentioned below) was inspired by the Purdue Reference Model (see Figure 5), which defines distinctive levels in manufacturing enterprises and functions. Furthermore, it includes details of functions in the control domain.

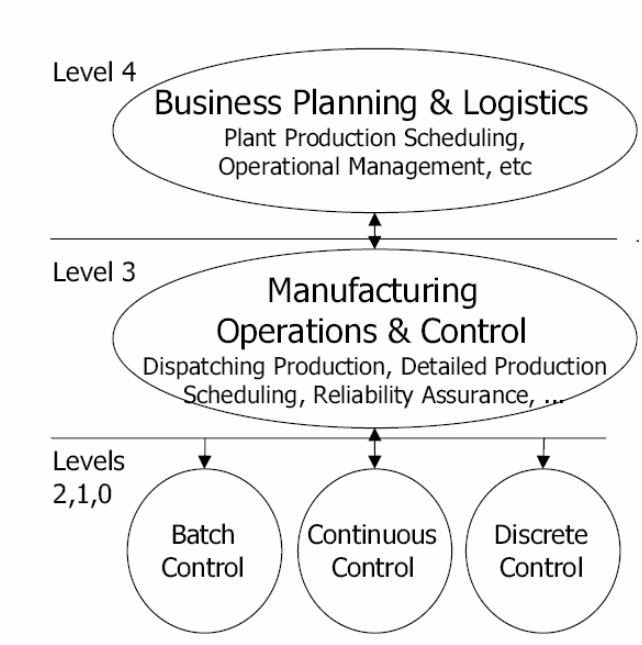


Figure 15. Multi-level hierarchy of activities (ANSI/ISA 2000)

ISA S95 aims at the standardization of interfaces between the levels of manufacturing environments (vertical integration, Figure 15 and between the manufacturing operation management activities within level 3 (Figure 16).

Still the interface Level 3/Level 4 is dominant in the specifications published so far (ANSI/ISA-95.00.01-2000 Models and Terminology; ANSI/ISA-95.00.02-2001 Object Model Attributes and ANSI/ISA-95.00.03-2005 Activity Models of Manufacturing Operations Management), but first considerations for internal integration of functional subsystems are undertaken in Part 3, which is therefore the standard with the greatest significance for our work: boundaries and behavioral description of functional entities, information flow identification, rudimentary data object definition. ISA 95.00.04 Object Models and Attributes for Manufacturing Operations Management and ISA 95.00.05 Business to Manufacturing Transactions still have draft status at the time of writing. Regarding Part 4 it will be interesting to see whether it is possible to find a common ground for object models not only for Level 3, but especially for field level automation interfaces, where a number of standards already exists (e.g. OPC XML Data Access). Part 5 will tangle the issue of transactions and hence message definitions, a field strongly occupied by the OAGIS 9.0 standard.

ANSI/ISA 95 is generic and at a high level of abstraction, thus it is representing a unifying framework which is highly adoptable for domain modeling. Nevertheless, it can cover the different engineering dimensions, but in a more user-friendly way compared to the preceding enterprise architecture approaches like GERAM or CIMOSA. This is partly the case because ISA S95 is considering the implementation restrictions, at least when it comes to the issue of modeling techniques. In that respect the IS modeling standard UML is used. It is the opinion of the author that a unified

modeling language is one of the key aspects towards mutual understanding. In this project it will be demonstrated that UML is extensible to support domain specific modeling, as well as to accompany modeling at different levels of abstraction.

ANSI/ISA 95 Part 1 defines hierarchy models, a high-level functional data flow model and an object model for the enterprise level and manufacturing level interface. This proposal incorporates a major weakness. The *ProductionSchedule* object assumes that all necessary information for every single *ProductionRequest* is cascaded into a single XML document. This may be convenient for monolithic and encapsulated systems in Level 4 (e.g. ERP system) and Level 3 (single MES system), but contradicts the idea of distributed functionality. To include the *MaterialConsumedRequirement* into the *ProductionSchedule* means to replicate static data (Bill of Material) within dynamic data (orders with time and resource constraints). Thus the proposed architecture just considers dynamic data and collects static data from defined service providers.

B2MML (Business to Manufacturing Markup Language) V2.0, 2003 is a W3C XML Schema Implementation for the ANSI/ISA standard, based upon the ANSI/ISA-95.00.02-2001 Enterprise-Control System Integration Part 2: Object Model Attributes Standard. The defined data object types are those specified for the manufacturing control functions and other enterprise functions interface contents (Level 3/Level 4 IF). That means B2MML helps to standardize the information exchange between business and manufacturing, but not between manufacturing functionalities. Unfortunately the element names in the XML type definitions do not always mach the corresponding ANSI/ISA 95 definitions, e.g. *SegmentResponse* in *ProductionSchedule*/*ProductionRequest* instead of *RequestedSegmentResponse*. In addition, for many objects the expression *ID* was included in the object names. Why these changes were made is not clear to the author and they eventually hamper the implementation due to the lack of compliance. Moreover, every object includes an *Any* element of the unbounded type *AnyType*, which allows for individual customization. This increases flexibility, and must be seen as a necessity for real world scenarios.

ANSI/ISA 95 Part 3 is a framework for integration projects. The focus lies on functions, not systems, organizations or individuals within manufacturing (Figure 16). The general problem regarding such a functional framework are the different IT and automation cultures and knowledge bases. In the past this resulted in a lack of consensus and of models to follow for integration. The increased need for such a consensus stems from demands like

- Supply chain optimization
- Asset efficiency improvement
- Agile manufacturing
- ATP (available to profit)

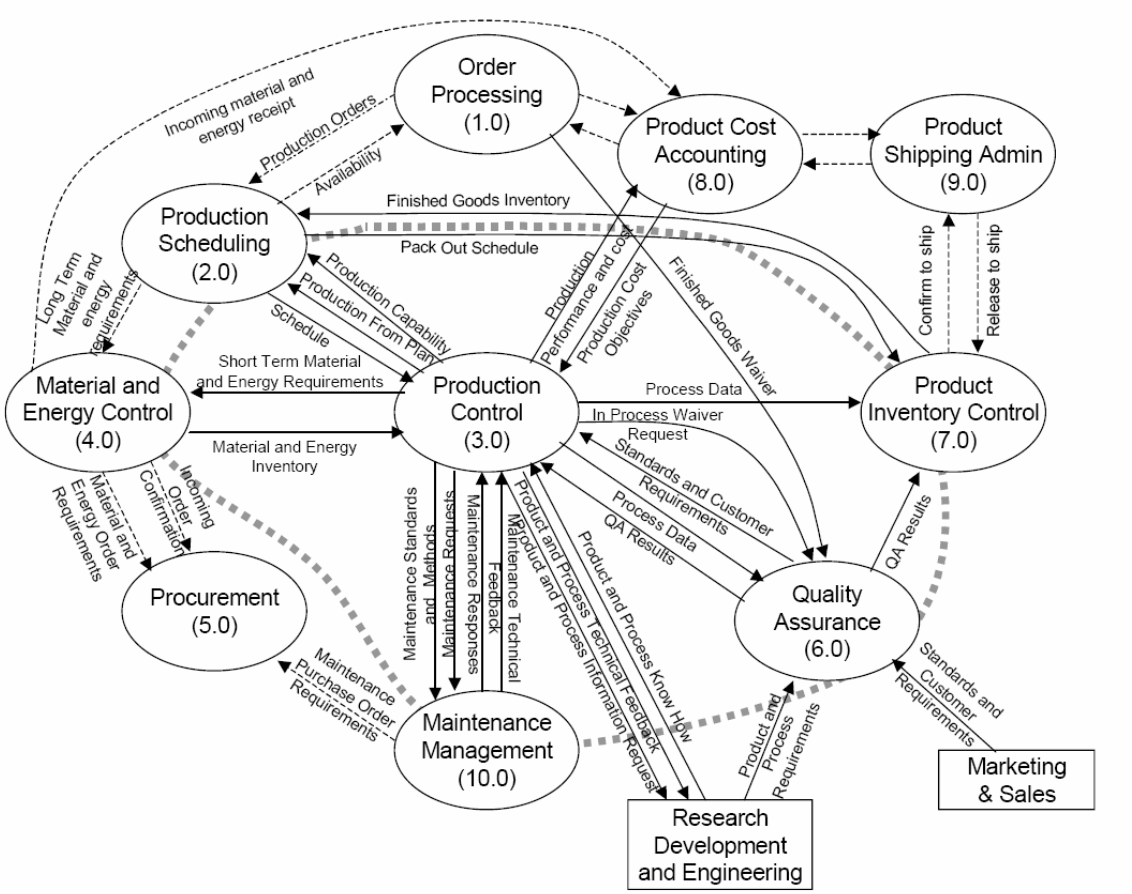


Figure 16. ANSI/ISA 95 Part 1 Functional enterprise-control model (ANSI/ISA 2000)

ANSI / ISA Part 3 tries to establish a consensus regarding activities of manufacturing operation management. As Figure 16 shows, at the highest functional abstraction level the connectivity between the key Level 3 and Level 4 activities is modelled in Part 1. Within this *Functional enterprise-control model* the border of the Part 3 *Manufacturing operation management model* is highlighted (dotted line). This line is the equivalent to the Level 3 / Level 4 interface defined in ANSI / ISA Part 1, which is also the source for the complete set of activities depicted in the figure above. Four categories constitute the manufacturing operations management model: production, inventory, quality, and maintenance operations management. The *Production operations management model* for instance includes the *Production Control* and part of the *Production Scheduling* function of Figure 16. Figure 40 depicts this relationship as part of the UML meta-model of the proposed methodology. The fact that there could also be other activities in a manufacturing facility is considered in Part 3 as well as in the proposed modeling framework. Within this project production operations management is the category which is used to implement the methodology as well as the use case.

# 3. Research Objectives

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. Single processes which seem feasible for the given situation are defined once during design-time according to the functionality the monolithic application provides. The system overview gets lost due to the restrictions of the application or interfaces. No matter whether an MES layer is introduced or there is direct interfacing between ERP and selected systems like tool management or scheduler, the problem stays the same: what is achieved is just process adjustment towards the functionality of the applications in a static framework and not vice versa (IT architecture follows process). This proceeding is not sufficient regarding the requirements of today's dynamic production environments.

But now more and more ERP vendors open up their monolithic products or develop new, service oriented solutions and offer coarse grained services via standardized interfaces. This creates new possibilities not only for static, data centric vertical integration, but also for full process integration of field and control level devices into service oriented architectures. Process logic could be designed to be that flexible so it renders time- and money-consuming, one by one interface coding obsolete. Present MES solutions suffer from this "interfacing-hell". At the end huge data integration efforts distract the focus from process integration. It can be assumed that sub-system vendors will follow the trend towards service orientation already visible at the ERP level, thus within the shop floor the distributed and standardized service availability will increase permanently. Although the three levels are still used, from a technical and functionality oriented perspective, hierarchies are avoided in favour of equal service providers and consumers. Service providers are at the enterprise level the ERP functionality blocks like material management (MM), production planning (PP) or finance (FI) as well as other business applications like MIS (Management Information System). At the field level PC-Terminals, SCADA or PLC can be mentioned, the future influence of RFID technologies was discussed above. Control logic at the automation level is distributed between intelligent automation devices. Interaction takes place via interfaces (lollipops and arrows in Figure 17) in a loosely coupled way. Thus process logic is not coded in one single application but can be flexibly designed and executed in an appropriate middleware layer. This vertical, internal integration of the shop floor is just one aspect, public services into and out of the shop floor the other. Machine vendors realize that it's not enough any more to ship the product without offering services like online maintenance (Wollschlaeger and Bangemann 2004), (Iung 2003), education support, planning and optimisation of production and logistics

systems or other e-Industrial / tele-services (Sihn & Graupner 2003). With material suppliers the integration is mostly established at ERP level and is mainly focused on procurement data exchange, although some procurement process activities could be delegated to the shop floor. For instance tool procurement: Today the demand triggered in the shop floor has to be entered into the ERP system. Then the purchaser processes the request and interacts with the supplier. Thus all the processing takes place at ERP level. A blanket order at the ERP level could be part of an efficient alternative arrangement which allows the worker in the shop floor to trigger automatically the process at the supplier side through supplier services whenever he enters the demand. Taking into account the growing flexibility for information process design, the shop floor architects have to have the preparation and equipment to integrate functionalities provided by their suppliers and also by their customers, who actually ask for available to promise functionalities of scheduling applications. Hence this kind of external connectivity at the shop floor control level is an issue at the customer side too. For instance track & trace requirements need actual product processing status data, thus not the ERP but the control level is able to deliver the proper data quality. In the opinion of the author the public, horizontal connectivity is in the short run not as promising as the vertical service integration due to higher coordination demands and security reasons. An architecture which supports regular process change and improvement is necessary. This would imply the implementation of a distributed architecture in the Shop Floor, with building blocks realizing the control and information flow in a loosely coupled way. Such an architecture, where functionality is not centralized in a monolithic application but is provided by decentralized entities through standardized interfaces would offer new possibilities, not only for static, data centric vertical integration, but also for full process integration of the machine level into service oriented architectures. If you take a look at Figure 17 you can imagine why facing this complexity without modeling based process life-cycle management is impossible.

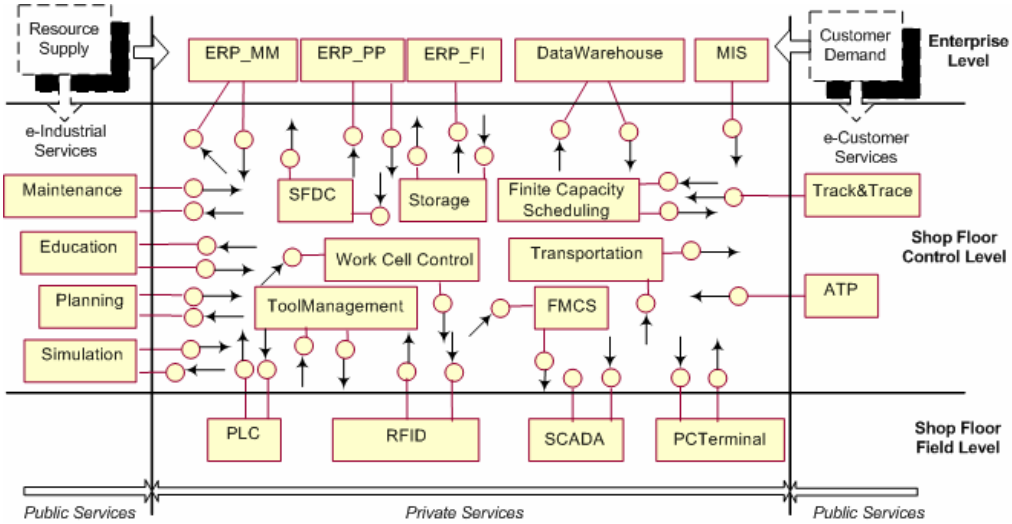


Figure 17. Internal and external service providers in the shop floor control level

Recent developments in and around the shop floor system strengthen the call for an intuitive, model based overview about service distribution and communication networks within the complete architecture. On the one hand more control tasks get transferred as close to the machine level as possible, because there the knowledge is concentrated and reaction times are shortest. On the other hand technologies will get influence which increase service distribution even more and thus the demand for consistent modeling.

> In the model-based systems engineering context, the research objective is a platform-independent UML shop floor domain model which contains a hierarchically ordered collection of services which are necessary to achieve typical, nested process functionality within different shop floor environments. A flow engine together with an appropriate middleware then executes this flow model. Therefore the modeling methodology has to support high-level system models as well as detailed, executable process models development. Business and IT alignment by means of modeling is a major aim, as well as the use of standards and best practices to keep the framework simple and make it user-friendly.

From a systems engineering point of view it is not surprising to see that the software engineering domain and the manufacturing domain are introducing similar concepts to cope with similar requirements their architectures have to fulfil. The markets (stakeholders) they serve demand:

- Functionality and Performance
- Flexibility (plug-and-play / produce)
- Reusability
- Reconfigurability, Customization
- Interoperability
- Portability

Their system elements (components) have to

- be distributed
- be loosely coupled
- offer easy, accessible and well-defined interfaces
- hide the implementation
- be context independent
- be self-contained
- be coarse grained
- be combinable

A manufacturing business and IT view alignment is a must for future manufacturing system life cycle management because:

- Ø IT is one of the enabling future technologies for manufacturing (e-manufacturing, internet based manufacturing)
- Ø Both worlds (Software Engineering and Industrial Engineering) can learn from each other regarding concepts and methodologies

# 4. Guiding Principles and Techniques for a Modern Shop Floor Architecture

Systems engineering for complex, socio-technical systems is only possible by means of:

- Modularization
- Standardization
- Modeling

Modularization is the capsulation of sub-systems and the provision of module functionality by means of well-defined interfaces. This approach is known for all kind of complex systems, thus also for information systems. The evolution of component- and object-orientated techniques finally resulted in the SOA concept, which strongly emphasizes the second point mentioned here: standardization. On the one hand SOA can be seen as an implementation-independent concept for modularized, distributed socio-technical systems. But on the other hand SOA also propagates a new programming approach strongly focused on standardized technologies, most prominently the Web Service stack. This is the major transformation step a feasible modeling methodology has to enable, the mapping of arbitrary functionality providers, be it software building blocks or humans, to the implementation platforms available. The latter can have an equal variation, from sophisticated complex Web Service interactions to paper flows. In the literature the two layers are also known as *Business Services* and *Software Services*.

In INTEROP D9.1 (2004) SOA, MDA and BPM are discussed in a holistic interoperability context, a fact which highlights the dependency of these approaches. Standardization not only at the technical level, but also regarding the semantics is closely related to the shift towards holistic approaches. Alonso (2004) speaks of domain dependent vertical standards, which glue together the business and IT level by means of shared terminologies and frameworks. For the Shop Floor domain one of the most interesting proposals in this respect are the ANSI/ISA 95 standards for enterprise-control system integration, which define hierarchy, activity and object models. Thus, a new information framework and architecture for the shop floor have to adopt these paradigms. In the following the system engineering principles used in this thesis are introduced.

# 4.1. Modularization - SOA

## 4.1.1. SOA Concept

Service Oriented Architecture (SOA)

> *… is a form of distributed systems architecture that is typically characterized by the following properties:*
>
> *Logical view: The service is an abstracted,* logical *view of actual programs, databases, business processes, etc., defined in terms of what it* does*, typically carrying out a business-level operation.*
>
> *Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.*
>
> *Description orientation: A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.*
>
> *Granularity: Services tend to use a small number of operations with relatively large and complex messages.*
>
> *Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.*
>
> *Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.*
>
> *W3C (2004)*

Below you will find a definition by Erl which emphasises the separation of SOA principles and the implementation by means of Web Services:

> *A SOA is a design model with a deeply rooted concept of encapsulating application logic with services that interact via a common communication protocol. When Web services are used to establish this communication framework, they basically represent a Web-based implementation of an SOA.*
>
> *Erl (2004)*

In this context Colan (2004a) defines a service in SOA as „... an application function packaged as a reusable component for use in a business process. It either provides information or facilitates a change to business data from one valid and consistent state to another. The process used to implement a particular service does not matter, as long as it responds to your commands and offers the quality of service you require." Berry (2003) states that "A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services." The last part of the statement is valid just for simple and not complex services which are choreographies of other services, that again are either simple or complex.

A SOA can be best described as a distributed, loosely coupled, standards-based and process-centred architecture, where the modules offer services, which can be accessed via messaging. Services in that context are discoverable (service broker publish and find interactions, Figure 18), stateless software entities that expose coarse-grained functionality by means of a well-defined interface. How this functionality is achieved, i.e. what happens behind the service interface, in which programming language the service is coded and on which platform the service is actually running, is hidden from the service requestor (loosely coupling). It seems that with this definition in mind it makes no difference whether the service is realized by a single method or a long-running process with user interaction. But the first case represents not the original aim of a SOA, which demands for coarse-grained services, in contrast to the fine-grained interactions of the OO world. Although the distinction between internal and external services will become less apparent, at the present stage external business services are often more coarse-grained than the fine-grained, internal services within a secure domain. To achieve enriched functionality, fine-grained services can be bundled through choreography (distributed process execution logic) or composition (centralized process execution logic). Beside reusability, this process centricity in terms of nested process designs is the key advantage of SOA. It enables real business process life-cycle management, that is to say fast process definition, process monitoring and continuous process improvement. In a SOA, the process logic is made explicit in flow models and not distributed between the service providers (Channabasavaiah et al. 2004).

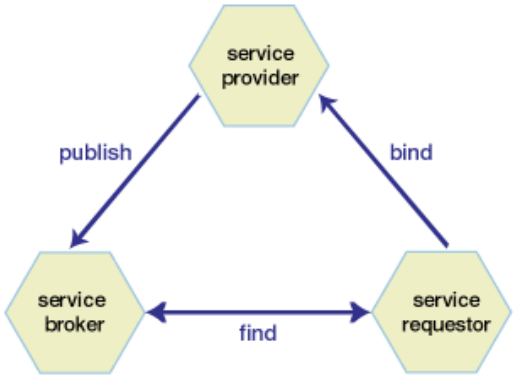


Figure 18. The basic SOA publish-find-bind mechanism

There is an ongoing discussion about whether SOA is a natural evolution of DOO (Distributed Object Orientation) architectures or a brand new paradigm. For us, when we assume an n-tier architecture, SOA is establishing the interface layer and can be seen in the application layer just as middleware providing transparent connectivity to the distributed pieces of application logic, which themselves apply to the OO paradigm. Hence there is a close relationship between the OO concept and the SOA concept. The OO concepts are already state of the art for single systems development, pushing forward the ISM techniques which are more and more necessary for analysing and design, and will become even more important in the context of MDA. At the shop floor level research with OO approaches was especially done for highly automated sub-domains like FMCS (Flexible Manufacturing Control System) or Manufacturing Cells, thus the work concerned a single and rather homogeneous platform system (Bruccoleri et al. 2003; Zhang et al. 1999; Booth 1998). For the design of MES applications OOA approaches still exist. Bauer (1995) presents the object-event-action model which is part of the ProduCASE tool from ABB. The problem is the usage within a unique and proprietary platform. The software building blocks offer homogeneous interfaces and interoperability is therefore not a real issue. In addition, the configuration of the processes takes place only once during the implementation.

As early as in 2000, when the SOA concept gained momentum, researchers warned that the lessons learned within nearly 20 years of OO should not be neglected when creating service oriented architectures. Burbeck (2000) states that "…how services are described and organized in a way that supports the dynamic discovery of appropriate services at runtime…" will define the success of B2B business. Architectures which focus on that issue he calls genuinely service-oriented, whereas such with a focus on message protocols and server communication details within a single corporate system he calls service-based. The author agrees on the statement about the importance of service categorization, taxonomy and discovery for the use during run-time and design-time, but the demo scenario in this work will demonstrate that the SOA application within a single domain does not make the service registry less important. On the contrary, today exactly the opposite comes true. In B2B scenarios the static point to point connection still prevails, whereas registries are successfully used internally.

The SOA again approves the concept of distributed computing, the platform independent modularisation of systems. Basis of this modularisation is the creation of encapsulated, reusable components which offer services through well-defined interfaces (not only IT-components can be seen in this way, but also the foreman, whose services include scheduling, exception handling, etc.). Self-similarity of the service providers regarding form and function is the basis and enable enriched usability. New systems make use of these reusable building blocks, which are also called "assets" in the context of software engineering (Colan 2004b) and will in many cases wrap legacy system functionality.

### 4.1.2. SOA and Data Distribution

The concept of encapsulation per se does say nothing about data distribution, i.e. which architecture is chosen for the data layer in a SOA. Generally speaking, two different approaches can be used:

- Ø Distributed approach: Persistent data is stored physically close to the source services or services which establish the highest access rate during the life time. Every service is offering data update operations and is responsible for updates in data dependent services.
- Ø Centric approach: Persistent data is stored in form of master data. A message router is responsible for the population of redundant copies in the services. Data changes are only allowed in the master copy.

The architecture described in clause two has the advantage of easier data quality control; also fewer components have to be highly available. The drawback is that routing related delays for data update have to be accepted, also the question, which data has to be updated in the services is not always easy to answer. Data updates in an architecture following the distributed approach trigger update processes invoking update operations of many services. One can find the opinion that a service encapsulating some database-resident data constitutes a business service. The only way to get to the data is via messaging.

### 4.1.3. SOA and System Management

The concept of SOA is not so much technology-focused, although the establishment of standards (e.g. WS at the interface level) is closely related to the further acceptance. It places much more emphasis on the importance of service life-cycle management, embedded in an overall system management. Similar to the technological record the SOA management concept is no innovation but derives from Business Process Management approaches. Out of the broad literature we will focus on the framework presented by Walford (1999), because he very much had IT supported processes in mind. The term “automation asset” that he uses for potential process elements can easily be equated with the term “service” used in the context of SOA. The formulated requirements toward corresponding management systems describe the SOA management requirements perfectly:

- Life cycle management
- Financial management (Accounting systems need to consider automation assets as “real” enterprise assets from a financial perspective.)
- Business rules (procurement, maintenance, utilization)
- Repository (information storage (metadata) and access mechanism)

#### 4.1.3.1. Life cycle management

The following are the requirements which SOA management systems have to fulfil in the different stages of the service life-cycle:

1) Creation stage management

This life-cycle phase can be further decomposed:

a) Need recognition (identified by means of insufficient operational performance in the service model or to-be scenario analysis requirements in the domain model)

b) Requirement determination

c) Rule based procurement:

- reuse assets the enterprise already has
- modify the assets the enterprise already has
- buy a COTS (commercial off-the-shelf) product
- modify a COTS product
- develop a custom asset in-house

d) Deployment (make available to potential user and register in repository)

2) Use stage management

Use stage management can be split into:

a) Configuration management

- versioning (evaluation of proposed changes to the product and version mix)
- updates
- interoperability
- aggregation / decomposition (service can be part of many processes, change in service requires effect examination in every process)
- multiple users (change management, update, keeping user information)

b) Operations management

The purpose of operation management is to maintain an environment that can effectively support the assets utilized by the enterprise.

- online management (monitoring, problem detection)
- offline management (activities which do not have to occur while the operational activities are taking place or that require a different time period in

which to accomplish the function e.g. preventative maintenance, help, repair, facility expansion)

- environment management (define a set of enterprise assets that must be managed as a unit)

3) Retirement stage management

Asset retirement has to be seen as a gradual process, and appropriate rules have to be enforced for decision making.

To achieve the outlined life cycle management approach, Walford (1999) differs between class models (system models in the context of SOA) on the one hand and unit models (service models in the context of SOA) on the other. Conformity between both models has to be ensured. Every service specification produced by the unit model needs to be checked for conformity to the system model and placed in its proper position in the model. The most useful location needs to be explicitly determined through business rules and experienced staff.

The CBDI, a SOA practice portal, postulates that

> *The Service Architecture provides the framework for managing the life cycle of services defining the policies and practices, infrastructure and common services for use in specific situations. … It helps answer questions such as:*
>
> *What services are required by the enterprise ecosystem?*
>
> *What services are available to the enterprise ecosystem?*
>
> *What services will operate together?*
>
> *What common semantics and business rules should apply to specific sets of services?*
>
> *What service usage should be standardized/mandated in particular circumstances? (security, management, CRM, product etc.)*
>
> *What substitute services are available?*
>
> *What are the dependencies between services and versions of services?*
>
> *Sprott (2004a)*

How this "crystal ball" has to look like remains unclear, because the suggested Business Service Bus, a "… logical view of the available and used services for a particular business domain, such as Human Resources or Logistics." (Sprott 2004b), depicted in Figure 19 is rather trivial.

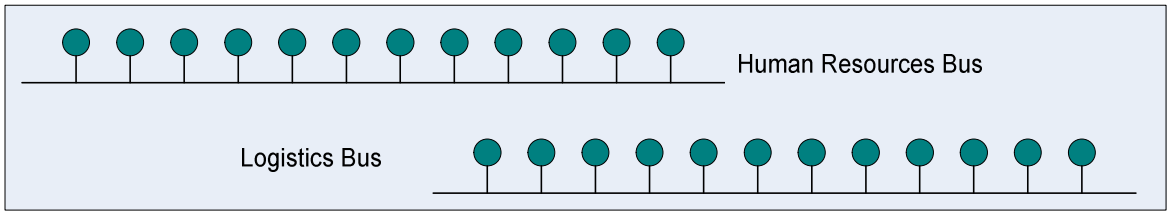


Figure 19.The Business Service Bus by CBDI

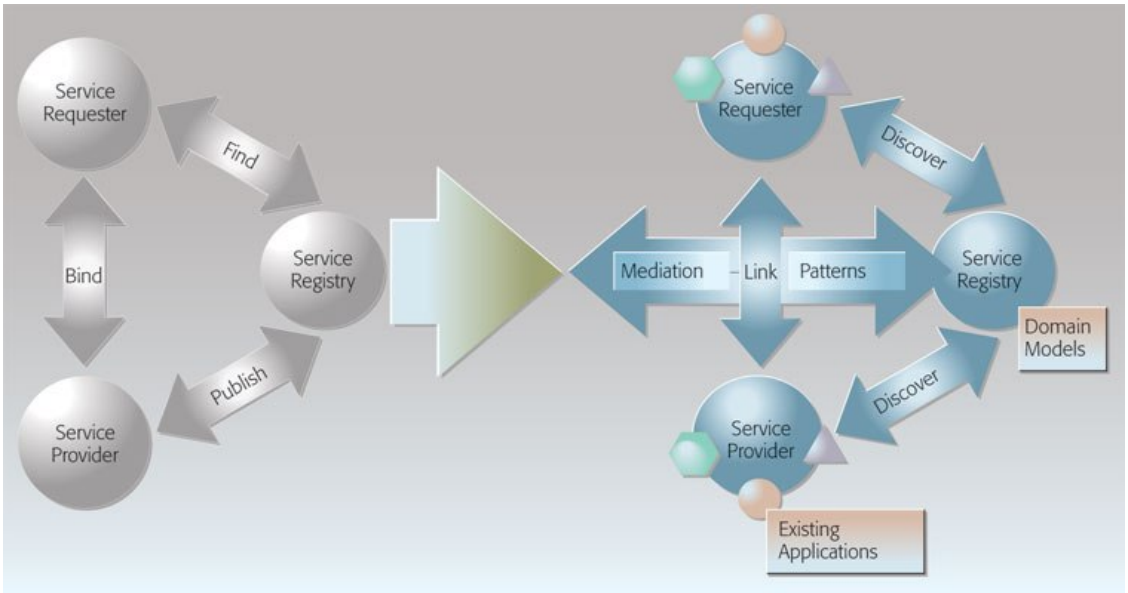


Figure 20. The extended basic SOA mechanism used in IBM's ESB (Schmidt 2005)

The idea of a *Service Bus* is widespread and not always as simple as the one by CBDI. IBM adopted this concept and calls it *Enterprise Service Bus* (IBM developerWorks Live 2004; Schmidt 2005); SAP extends the idea towards an *Enterprise Service Architecture*.

In (Schmidt 2005) the importance of the registry as a means of metadata (interface description, policy) storage is singled out within the ESB integration model, which is explained by means of high-level conceptual UML models (*Domain Models* in Figure 20). But the actual integration of a domain model in a SOA is not further detailed.

One of the aims of this thesis is to give an elaborate example how model driven service architectures can be realized and managed. The goal of such an approach is still the same as defined above, that is to say support for critical decision making processes (around standardization of scope, usage and acquisition or how existing applications support the new service architectures).

#### 4.1.3.2. Financial Management

Except in the Software Engineering business, where the software components are the core assets, this point is still underrepresented in companies with high IT penetration. Even more stunning is the fact that in the WS-SOA discussion this aspect is rarely considered, although, as already mentioned above, the original unique selling position for this concept was seen in the B2B and B2C context.

Due to the operational focus of this work financial considerations concerning service assets are not included. Nevertheless, the pricing of services and therefore accounting for SOA will become an important issue not only for inter-organizational scenarios, but also for cost transparency in an intra-organisational business context.

#### 4.1.3.3. Business Rules

Business rules like trigger conditions help to make many policies and procedures explicit and promise to

- enable non-technical personnel to define the operating principles (e.g., control, algorithms, policies, procedures) utilized in business automation functionality (applications)
- enable the operating principles to be changed quickly without the need for programming or reprogramming applications
- enable the analysis of the operating principles from a global perspective to determine if they are complete, consistent and non-conflicting
- ensure that different applications utilize the same operating principles that are needed for the functionality involved
- ensure that the operating principles are in conformity with the desired way for the enterprise to operate

Business rules are statements that articulate an operating principle by defining or constraining some aspect of the business. According to Walford (1999) they must

- be able to be followed
- be consistent
- be able to be articulated
- be able to have their compliance measured
- have constraints

He also suggests a classification into three strength classes (requirements, standards and guidelines). Rosenberg and Dustdar (2005) use integrity, derivation and reaction rules and present a loosely coupled approach to the way of implementing these rules in BPEL processes. They contrast this approach with a tight coupled approach, characterized by a direct interfacing between BPEL engine and business rule engine via a proprietary API. They are absolutely right when they complain about the lack of reusability of the latter approach, because the rules should be applicable for all kinds of applications as a service. Still, also the proprietary API should be seen as a service, which can be reused at least by different processes running at the communicating process engines. As will be shown in our demo scenario, the lack of service orientation can be minimized by means of wrapping the proprietary rule call into a parameterised service (in our case a BizTalk orchestration which can be deployed as a Web Service). Rosenberg and Dustdar have extended this wrapping mechanism and call it a Business Rule Broker, allowing a unified access to different BRE through a Web service interface. Moreover, not all kind of rules demand for reusability and those for loosely coupling, e.g. business rules for flexible process configuration are tailor-made and hardly reusable.

#### 4.1.3.4. Repository

Repositories have to support publish and discovery mechanisms in the basic SOA paradigm. Beneath that meta-data for service specification should be provided to ease service organisation. Generally speaking different occurrences of repositories are feasible, centralized, decentralized or hybrid. This comparison will be continued when we talk about directories and UDDI (Section 4.2.6).

## 4.2. Standardization - Web-Service Architecture and ANSI/ISA 95

We have to mention the Web Service Technology, because it is one of the main forces behind SOA. Its main advantage is the standardized connection mechanism with XML and SOAP. The existence of technical interface standards for systems integration is crucial regarding interchange ability and reusability. In the context of SOA loosely coupling indicates this demanded flexibility. Greenstein and David (1990) define technical interface standards as "… the collection of explicit rules that permit components and subsystems to be assembled in larger systems and hence are also called technical compatibility standards." (cited from Steinmueller 2003).

But the WS-Architecture is just one example for a SOA. DCOM, CORBA (Sprott 2004a) or proprietary connection and messaging capabilities must be seen as direct predecessors and equal technologies in service oriented enterprise architectures. Of course one has to have in mind the possible implementation techniques, but more important for us is the question whether the SOA and the MDA concept together can help to overcome the weaknesses of the actual shop floor structures or not.

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP- messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*
>
> *W3C (2003)*

Web Services are:

- Loosely-coupled (because in general they are defined, developed and managed by different companies)
- Autonomous and independent (decoupling of applications makes them more modular and enables reuse and aggregation not known at design time)
- Software applications with published and stable programming interfaces
- Exposing the functionality performed by internal systems and making it discoverable and accessible through the Web in a controlled manner

- Homogeneous wrappers (a wrapper is a program which acts as an interface between the caller and the so called wrapped code. Wrapping is an OO engineering pattern as well), in that they interoperate through Web standards (e.g. HTTP, XML)
- Establishing an additional integration layer and therefore overhead mainly through message conversion, which demands more coarse-grained operations

Web Services are not:

- an integration solution by themselves
- replacing the need for middleware and many traditional integration technologies, at least concerning the internal architecture (conventional middleware for multi tier architectures hidden behind the WS wrapper and WS middleware for message transport and processing (e.g. HTTP and SOAP server))
- the best choice if performance really counts, especially in tightly coupled, high-volume internal applications due to the XML parsing and serialization overhead (Brown 2003)

Loose coupling usually involves some degree of penalty in terms of 'performance', although it is important to distinguish between engineering and economic performance. Engineering performance relates to the technical performance of the system, for instance point-to-point message delivery times. Ng et al. (2005) tested the latency and throughput performance of some current commercial SOAP implementations and compared it with Microsoft .NET Remoting. Concerning latency, the non-SOAP .NET Remoting using binary encoding and direct TCP/IP transport performed better than the fastest SOAP implementation (Document/Literal encoding style), although the difference between small (appr. 0,9 kb messages result in a 1:4 relation) and complex (appr. 20 kb message perform nearly equal) messages has to be considered. Due to the fact that in the shop floor control level opposed to the field level complex messages prevail, SOAP may still meet the performance requirements of this application. Regarding throughput (message/seconds) .NET Remoting performed up to 20 times better. This indicator can be completely ignored because the interactions between the shop floor control level services are far from this throughput levels.

Now that it is clear that the WS/SOAP interface standard will do well for the domain under discussion, the question is how to get familiar with the WS technology stack as described further down. Barry (2003) mentions three distinctive steps to depict the impact of WS regarding EAI:

- Ø Improving Web site connectivity (e.g. gadgets that can be added to Web Sites delivering sophisticated content from different sources (e.g. Stock Quotes, News, etc.). This approach was further improved in the portal development.)
- Ø Improving internal connectivity
- Ø Improving Business-to-Business connectivity

Through the strong industry support and the high standardization level Web services are promising wrapper technologies in the context of EAI. Whereas the ultimate goal of WS remain inter-company interaction, most applications today are deployed within a single company LAN (point 2), often similar to a client server model. WS are used as platform independent integration technology for heterogeneous (legacy) systems. In using SOA, one can expand the use of Web Services from simple client-server models to systems of arbitrary complexity (Colan 2004b). The point 3 perspective was strongly misused in the beginning of the WS hype, which led to a number of "very ambitious" forecasts. Let us take a look at one of the well known GARTNER statements, postulating that

> *No later than year-end 2006, 90 percent of application development staffs in the Global 2000 will be developing secure, marketable services for external use (0.8 probability).*
>
> *Andrews (2003)*

The reality shows that there are a number of mature external, public WS available (e.g. Amazon, Google), but the promise of externalized functionality offered at a service market as a business model has yet to be held. Moreover, the vision was prevailing that dynamic partner selection for WS interaction will succeed once a WS market is established. Public registries like UDDI could become the hub for partner assembling at run-time. In fact the public UDDIs of Microsoft or IBM soon were useless because of the high number of low-quality services registered, and if WS are used for B2B interaction today, the partners are selected at design-time and well known. The enhancement of the WS stack regarding security or policies is aimed to overcome the insufficiencies for B2B scenarios, the implementation of these second generations WS is still at the beginning and a final judgment on this matter is not possible at this point.

A detailed picture of the WS-SOA stack offers Figure 21 found in ebpml.org (2003), but an exhaustive discussion of all technologies involved is out of the scope of this work.

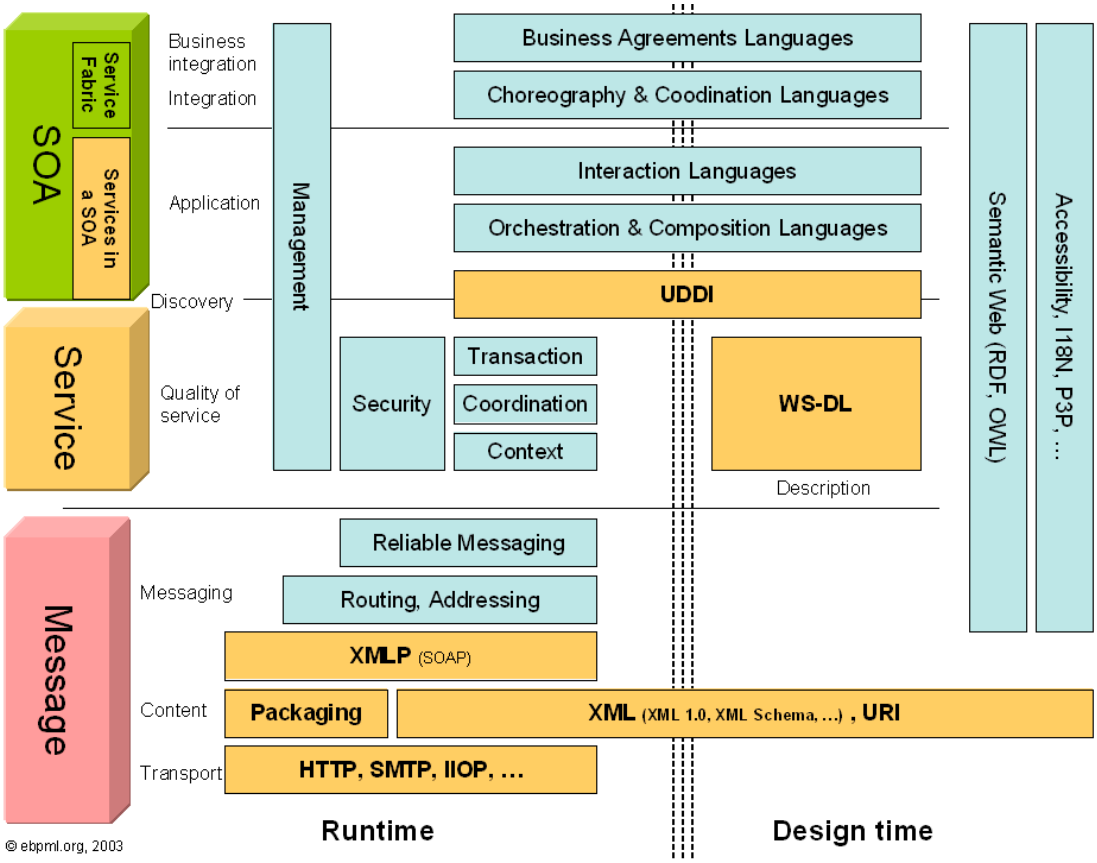


Figure 21. Extensive WS-SOA technology stack (ebpml.org 2003)

For the introduction into the related technologies for WS-SOA realization we follow the proposal found in (Alonso 2004). The service description and discovery stack introduced there is depicted in Figure 22, enriched with the technologies actually used in this project. The messaging layer technologies regarding transport (SOAP over HTTP) are not further discussed. Reliable messaging and routing is not essential for us due to the proposed architecture where a central broker (BizTalk Server) is responsible for message routing.

Hence, the mainly centralized control architecture with point-to-point connections reduces the requirements at the protocol level. In addition, the intra-organizational focus spares much of the security issues so critical in inter-organizational service flow scenarios.

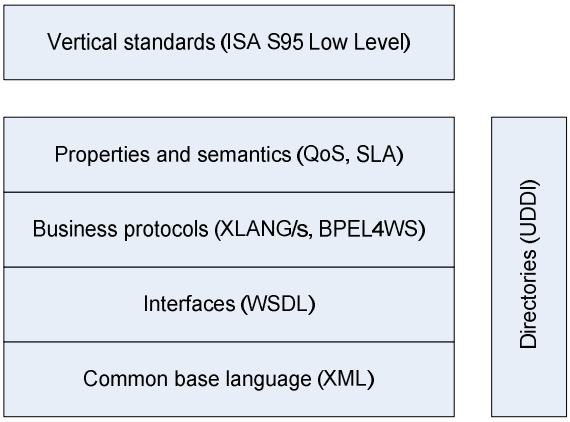


Figure 22. Technology stack for service description and discovery (adopted from Alonso (2004))

### 4.2.1. Common Base Language

XML is used for both, as a common meta-language for specifying many other service description languages related to SOA (e.g. XML Schema (XSD) in Figure 23) and as the preferred data storage and message payload format within the SOA. All specifications depicted in the picture are under W3C (World Wide Web Consortium) custody. The widespread industry support and its flexible syntax are the strongholds for this language. Most relational databases provide some form of XML syntax import and export functionality today. With XML the syntax of the message payload is given, but the semantic remains open. Hence there are many, mostly industry related proposals under discussion. OAGIS (Open Applications Group Integration Specification) provides the definition of business messages in the form of Business Object Documents (BODs) and example business scenarios that provide example usages of the BODs. OAGIS BODs can be included as SOAP message body parts. IBM has adopted the OAGIS 8.0 specification to develop an enterprise messaging architecture and standardized message vocabulary that can be used across the corporation for application integration, called EIMS (Enterprise Integration Messaging Specification). The aim is to standardize the message payload within the SOA by means of generic business object documents.

In the demo-scenario all data exchange will take place in XML format. Central stored XSD (version 1.0) files are used to describe the data types. Persistent data is stored as XML files (mainly message archive functionality), as Excel files (e.g. Product Definition Data) or in ASCII file format (Production Requests in the Scheduling application). XQuery (actual version 1.0) is a specification for XML documents searching. This is used heavily in the BizTalk Orchestration Designer Expression shapes which provide direct message browsing functionality. XQuery uses XPath (version 1.0), a specification which introduces a method to describe XML document syntax through path expressions. XSLT (Extensible Stylesheet Language Transformation, version 1.0) transforms XML documents, again utilized in BizTalk for data type mapping. The complexity of a transformation is hidden by means of a graphical tool.

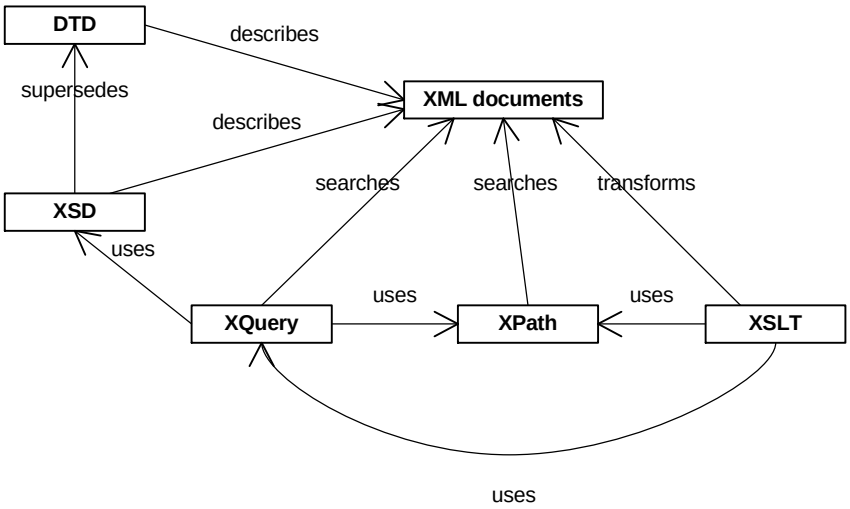


Figure 23. Relationship between XML specifications (Erl 2004)

### 4.2.2. Interfaces

> *WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).*
>
> *W3C (2001)*

As with the XML specifications, it is assumed that the reader is familiar with the basic Web Service technologies. WSDL (Web Service Description Language) in the latest version 1.1 is a core element of the WS stack, allowing service specification description.

### 4.2.3. Business protocols

There are a number of XML based standards for process markup languages available, most prominently BPEL4WS, which has its roots in Microsoft's XLANG and IBM's WSFL (Web Service Flow Language) specification. Competing standards are BPML (Business Process Markup Language), defined by BPMI (Business Process Management Initiative), or XPDL 2.0, released October 2005 and hosted by WfMC (Workflow Management Coalition). The first has had little support from the very beginning; the latter still lacks the support of the big players in the execution engines market, although smaller vendors do have compliant workflow engines. Others support at least XPDL import/export functionality.

> *The Business Process Execution Language for Web Services (BPEL4WS) provides a standard way of describing business processes that are based on Web services. Existing standards, such as WSDL, do not provide facilities that are required in complex business protocols, such as the description of behaviour dependent on data sent between services, exception management and recovery, and coordination between business partner participants for long running and complex processes. BPEL4WS aims to meet these requirements.*
>
> *Endrei et al. (2004)*

BPEL4WS was first specified in July 2002 by BEA, IBM, Microsoft, SAP, and Siebel. The specification was constructed to combine the approaches previously described by Microsoft (XLANG, for the BizTalk server) and IBM (WSFL, for the WebSphere server). The latest release of the specification at the time of writing, version 1.1, was submitted to the OASIS standards group in May 2003.

### 4.2.4. Properties and Semantics

Semantic issues must be differentiated in design time and run time requirements. With the repository both domains can be tackled. Classification and tModels in general are suited to support design time decisions as well as run time service selection. tModels are used in UDDI to point to the endpoint of an arbitrary selectable document and saves this reference under a unique key. Service descriptions contain mostly a tModel pointing to the corresponding WSDL document. For instance, in the proposed demo scenario tModels describing the service state (test, production …) are introduced. Such a specification of properties and semantics at a service level is not sufficient, because it can only provide limited information about the service and its overall functionality in the architecture as well as in interaction with collaborating services. Nevertheless, at run time the UDDI service information is the only one available in a centralized architecture concept. The approach presented here is to provide system context semantic by means of models and a modeling framework. The final aim is to use the models to establish mutual understanding about the functionality in the system during design time.

### 4.2.5. Vertical standards (Manufacturing Domain)

Robert Mick of ARC Advisary Group, which call themselves “Though Leaders in Manufacturing and Supply Management”, states that

> *… integrated, flexible IT systems that eliminate the boundaries between enterprise and manufacturing applications are vital for manufacturers that are striving to synchronize their operations and supply chains. But developing such systems can be challenging given the high degree of system diversity between these two domains and the traditional independence of corporate and manufacturing IT planning. Interest in enterprise architecture and standards-based interoperability that can bridge these gaps is therefore quite high.*
>
> *Mick (2005)*

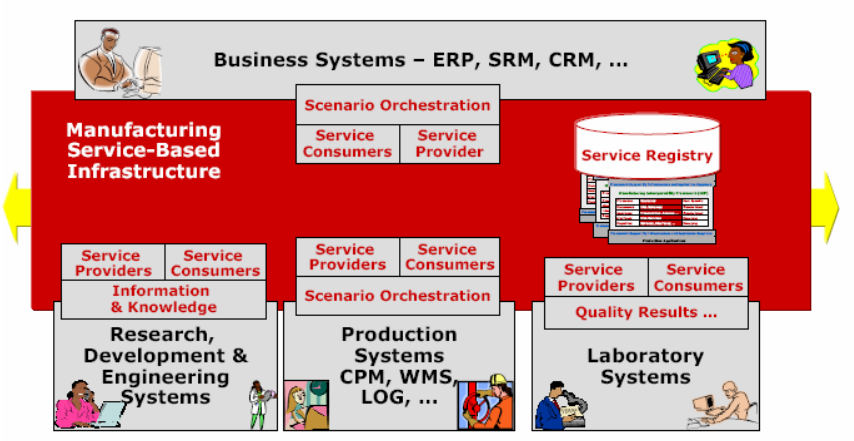


Figure 24. Proposal for a Manufacturing Service-Based Infrastructure (Mick 2005)

Figure 24 depicts his proposal for a Manufacturing Service-Based Infrastructure to realize standard based vertical integration following ANSI/ISA 95. Although containing some very true assumptions, it is unfortunately typical for existing proposals in the sense that it remains at a high level of abstraction and therefore unsubstantial. Such proposals are neither formalized, nor do they consider implementation environments. In his opinion an enterprise architecture combined with strategic system management is needed to achieve integration at an application and physical network level, and mentions Service Based Architecture (SBA) as an example. But only a SBA "… combined with standards-based interoperability and an environment that is conducive to rapid change." will leverage the architecture's potential regarding flexibility and adjustability. All the more, Grid computing is relying on service orientation as well. WS-SBA offers standards just at the interface level, but more standards regarding "… documents, actions, transactions, scenarios or business processes." have to supplement WS standards. The noted OPC Unified Architecture is depicted as a service-based proposal trying to provide "… manufacturing operations with a unified data-model … ". In the following he argues in favour of a role-based, human-centric approach which makes extensive use of libraries of re-usable services. He concludes with the recommendation of a collaborative approach regarding corporate and IT planning as well as interoperability standards and architecture.

So far the potential of ANSI/ISA 95 as a vertical standard was recognized, but a concise approach how this standard can be integrated in a model driven SOA concept is missing.

### 4.2.6. Directories

Dustdar and Treiber (2005) undertook a view based analysis of Web Service registries. They compare centralized, decentralized and hybrid architectures for registries and discuss the pros and cons of UDDI (Universal Description, Discovery and Integration) as a centralized approach. They also separate the human view from the WS view, a

proceeding which is in line with the different roles a registry has to fulfil either at design time or at run time. While the author totally agrees with most of their statements (regarding fault tolerance, management), the criticism of the narrow scope of the UDDI data model (attribute extensibility of data model) is hard to follow, because like in ebXML in UDDI registries arbitrary data can be referenced via tModels and thus assumes no limitations about the content being stored just as well. Also the absence of semantic metadata was in principle correctly identified, but the categorization mechanism is a proper tool for incorporating metadata into the registry. We will demonstrate how this was achieved for the shop floor semantics. The correctly described cumbersome query mechanism of the UDDI API could be improved by means of an UDDIHelper class, which hides the complexity of queries from the requestor.

Due to infrastructural restrictions and the need for a ready to use solution, UDDI was chosen for the proposed scenario. Replication is not considered in this scenario, not at least due to the restrictions of UDDI 1.0. UDDI 2.0 offers enhanced replication mechanism between multiple registries, but could not be considered at the time the implementation took place. Further work would definitely have to consider this issue.

Even if a detailed discussion of UDDI is beyond the scope of this work, some hands-on papers shall be introduced which provided valuable insights in how to work with UDDI. Januszewski (2002a) highlights the importance of meta-data and demonstrates how to create custom categorization schemes. Brittenham et al. (2001) examine the different methods of using WSDL with UDDI registries. A key concept regarding WSDL files as tModels in UDDI is to delete the *service* tag from the description, because the end point is stored as an UDDI Binding Template. These were examples for design time considerations. How UDDI can be leveraged at run time is described by Januszewski (2002b).

## 4.3. Modeling

### 4.3.1. Modeling Related Problem Spaces

Modeling of processes is a widespread accepted instrument for organisation (e.g. Business Process Reengineering (BPR)) or application (e.g. ERP system implementation) design. Until recently existing modeling approaches in the context of (IT supported) business processes had major disadvantages.

Model usage intensity

As already mentioned, the concept of modeling for a better understanding of the structures and processes in the enterprise is not new. But while the demand for modeling is in theory accepted for system change or improvement, the reality is different, as the survey of Whitman and Huff (2001) shows. They observed in their survey poor use and therefore updating of existing models. A reason for this is that

modeling is often seen as a unique task, which supports the process design once, not in a permanent process of change and improvement.

### Organizational framework

An additional handicap for model driven decision making can be found in the widespread, strongly hierarchical structure for both, organisation and system architecture in the enterprise. The horizontal organisation, despite all process thinking efforts, is still determined by the departments and their functionality, like Porter designed it 20 years ago. In the vertical dimension a layered architecture is postulated, with the shop floor often as a black box layer due to complexity and variety.

While in the horizontal dimension of the management layer the single ERP system dominance led to a more or less similar process understanding and a rather high degree of (potential) workflow automation, the shop floor layer with its heterogeneous environment of men and machine controls demonstrated that the IT support for business processes is not unlimited. In the ambitious BPM visions the shop floor can be found as one entity in a multi-enterprise supply process, but in the IT-reality time consuming and expensive interfaces have to be programmed. So not only the gap between business desire and IT reality is obvious, but also the different levels of abstraction, where BPM tended to remain at a high level compared to the ISM issues.

### Modeling Techniques Variety

This given organization has always been a rigid framework for process design, which was in the early days a pure management issue, but due to increased technological feasibilities soon IT became the enabler for process automation. So historically two more or less separated paradigms have evolved, the business and the information system view, resulting in two groups of modeling techniques, one for BPM (Business Process Modeling) containing IDEF0, Petri Nets, ARIS, Flowcharting, etc., the other for ISM (Information System Modeling) containing Data flow diagramming, ER diagramming or UML. Integrated design strategies rarely have been the case in practice (Giaglis 2001). The difference in modeling techniques has not only widened the gap between the business analysts and the IT-specialists in the way they build up their architectures and design processes. Haeckel (2003) goes one step further and postulates that most existing models are created by IT-architects, thus not representing executive intent. He claims that the business architect counterpart is missing because business people do not think like system designers at all. They have just learned how to design processes. Walford (1999) states that

> *One failing of many process reengineering and management-by-process efforts is that the entire focus is on process definition and very little attention is paid to the process implementation needs and the associated life cycle management requirements.*

He continues that

> *In many, if not most, businesses, the definition [of processes] is informal and has grown and changed throughout the life of the enterprise with little or no attentions. ... Most business have been quit successful over a long period of time without a define approach. This was possible because of the inherent constraints imposed by the hierarchical*

> *organization and function-based automation support. Those conditions, to a great extent, mitigated the lack of formalized models. With the change to a relative flat, process-based organization and client/server distributed automation support, the lack of a planning model puts any enterprise at a considerable disadvantage. The disadvantage is external with respect to competitors who are able to model and understand their business. The disadvantage is also internal, in that the efficiency of operation will be far less than it could or should be with a well-thought-out model.*
>
> *Walford (1999)*

In the opinion of the author the external disadvantage is somewhat overstated, because the success of the companies without models proves that they understand their business. But there is no doubt that the increasing connectivity in an increased global network organisation can be better handled by means of a business model. Furthermore, and this is driven through the increasing demand for external interfaces, a consistent mapping between the business model and the implementation model is crucial.

Limited scope

As we will try to make clear in the following, to fulfil actual requirements concerning operational model use enlarged methodologies are needed. For most of these scenarios a permanent semantic actualisation of the models was not necessary up to now. Either these models have been per se used only once (BPR), or they have been stored for documentation purpose only (ERP reference models, ISO certification). Thus they remain process models instead of architecture models, which in contrast would visualize not only a single, but the sum of all possible processes. Nevertheless at the enterprise level these limitations were already recognized and business process life-cycle management starts to influence modeling behavior (Becker et al. 2003).

### 4.3.2. Model Driven Architecture

Modeling in the context of information systems lifecycle management has one major driving force today: knowledge alignment. If one's aim is real process flexibility, then the model has to support the process user in choosing between alternative flows or changing control and information flows. The users should interact with the system at the same level of abstraction as the domain analysts, who initially set the static and dynamic structure. Both roles demand simplicity (Haeckel 2003). The models should also absorb the system and IT knowledge about platforms, and finally the programmer knowledge about development practices. Concerning dynamic system behaviour, which is the focus of this work, the stakeholders' diverse backgrounds reflect a major challenge. For one group, process modeling is an essential part of overall system modeling, for the next it is just a graphical vehicle within the IDE and for the third it is a pure visualization of their business logic without any IS context. The strong focus is on architecture dynamic and adjustability, together with an ever growing analogy between business- and IS-processes, that is to say the IS has frictionless control over the process, making a common view necessary. To achieve this, generally speaking, two main model driven process design and execution

approaches exist. The first one postulates the concept of model enrichment through extensions combined with elaborate transformation and mapping mechanisms, most prominently represented by means of OMG's MDA (Model Driven Architecture) (Miller and Mukerji 2003).

> *MDA provides a set of guidelines for structuring specifications expressed as models and the mappings between those models. The MDA initiative and the standards that support it allow the same model specifying business system or application functionality and behavior to be realized on multiple platforms. MDA enables different applications to be integrated by explicitly relating their models; this facilitates integration and interoperability and supports system evolution (deployment choices) as platform technologies change. The three primary goals of MDA are portability, interoperability and reusability.Portability of any subsystem is relative to the subsystems on which it depends. The collection of subsystems that a given subsystem depends upon is often loosely called the platform, which supports that subsystem. Portability – and reusability - of such a subsystem is enabled if all the subsystems that it depends upon use standardized interfaces (APIs) and usage patterns. MDA provides a pattern comprising a portable subsystem that is able to use any one of multiple specific implementations of a platform. This pattern is repeatedly usable in the specification of systems. The five important concepts related to this pattern are:*
>
> *1. Model - A model is a representation of a part of the function, structure and/or behavior of an application or system. A representation is said to be formal when it is based on a language that has a well-defined form ("syntax"), meaning ("semantics"), and possibly rules of analysis, inference, or proof for its constructs. The syntax may be graphical or textual. The semantics might be defined, more or less formally, in terms of things observed in the world being described (e.g. message sends and replies, object states and state changes, etc.), or by translating higher-level language constructs into other constructs that have a well-defined meaning. The optional rules of inference define what unstated properties you can deduce from the explicit statements in the model. In MDA, a representation that is not formal in this sense is not a model. Thus, a diagram with boxes and lines and arrows that is not supported by a definition of the meaning of a box, and the meaning of a line and of an arrow is not a model—it is just an informal diagram.*
>
> *2. Platform – A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.*
>
> *3. Platform Independent Model (PIM) – A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.*
>
> *4. Platform Specific Model (PSM) – A model of a subsystem that includes information about the specific technology that is used in the realization of that subsystem on a specific platform, and hence possibly contains elements that are specific to the platform.*
>
> *5. Mapping – Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel. A mapping may be expressed as associations, constraints, rules, templates with parameters that must be assigned during the mapping, or other forms yet to be determined."*
>
> *OMG (2003)*

Process Automation can be seen as a layer stack containing

- system model layer
- process layer
- communication layer
- IT-layer

In the past more than once only the latter two were considered. Communication demands lead to IT decisions; integration efforts are strongly driven by IT issues and are not business-model-based. In a SOA context, the first two layers could also be seen as part of the business layer, then the service layer and least the component layer (Zimmermann et al. 2004).

The MDA postulates a tight coupling between model and code allowing as far as possible automated creation and synchronization in both directions, from model to code (forward engineering) and from code to model (reverse engineering). This aim calls for a common modeling technique for both, business analysts and IT-architects. Within the MDA this unifying language is UML, the standard modeling technique for software engineering. The concept is originally based on a three tier architecture, with a Platform Independent Model (PIM) at the top, the Platform Specific Model (PSM) in the middle and the applications at the bottom. Due to the real-world demand for models which make no assumption about the implementation environment, a fourth layer was introduced, namely the Computation Independent Model (CIM). Figure 25 depicts the principle proceeding.

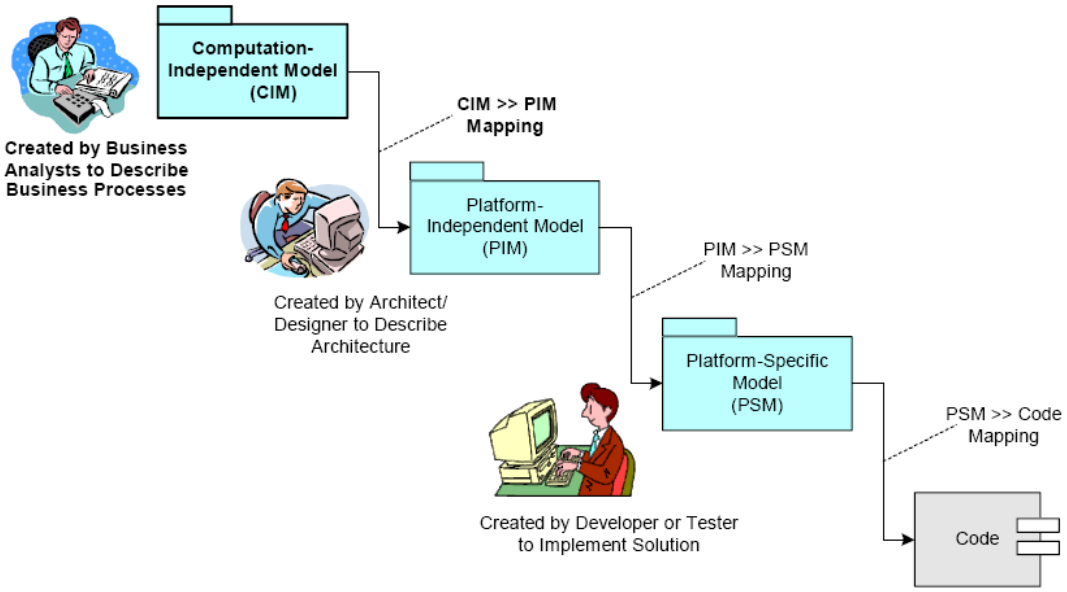


Figure 25.Model Driven Architecture abstraction levels (OMG)

What has to be criticised here is the usage of a waterfall approach, where no feedback loops are foreseen. Reverse engineering, the reverse mapping between code and PSM is an example which works at least for static information (e.g. class definitions) very well. To succeed in a dynamic environment of system life-cycle management, bi-directional mapping between all MDA levels is an imperative as well as a challenge. The methodology presented in this work will consider control loops not in the form of formal mappings as defined above, but for the operational management of executed processes.

CIM and PIM are consistent with appropriate core models which represent specialized environments. PIM and PSM are modelled with UML. The benefit is a stringent process from abstract domain models down to executable code. Therefore, it is called a "top-down" approach.

The second model driven process design and execution approach is best described as "bottom-up", because not the portable architecture model is the starting point, but the existing process development and execution possibilities within a certain vendor environment. The vendor independent visualization is just an extension of graphical drag & drop interactions offered by the IDE. In this group we classify MS Biz Talk 2004/Visual Studio 2003, IBM WebSphere Studio (Kloppmann et al. 2004) or the SAP NetWaever Platform (Woods & Word 2004). In addition to platform specific (PSM), the IDE models have to be called vendor specific (VSM). Only very recently within this group portability is not only achieved through the mapping to process mark-up languages like BPEL4WS, but also at model level through e.g. UML-XMI mapping (Beck et al. 2005). Both paradigms have the above mentioned knowledge alignment in mind, but the difference in the realization shall be described by means of two examples for model driven Web Service Orchestration.

It is important to understand that process life cycles are separate and distinct from SW life cycles, but they interact closely. First, processes could be implemented completely by human effort. Second, in a process that requires SW support because of advancing technology, several generations of SW, each with its own life cycle, could be employed without changing the process. The MDA approach includes this fundamental principle in so far as this approach separates computer independent, platform independent and platform specific models.

# 5. Model Driven Service Architecture for the Shop Floor Domain

The management of control and information flows in the shop floor is a demanding task. The reasons for that have been outlined in Section 2. Due to the weaknesses of existing solutions, the aim was 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 the concept of a MDSA (Model Driven Service Architecture) for the shop floor. It is 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.

## 5.1. Related Work

As said before, the EA proposals provided some useful information about modeling frameworks, but performed poor at the implementation level. In the MDA context, our research objective is a platform-independent UML shop floor domain model which contains a hierarchically ordered collection of services which are necessary to achieve typical, nested process functionality within different shop floor environments. This model has to be mapped on the actual, platform specific service assets available, creating flexible service flows. Leymann and Roller (2002) present a similar approach called *flow composition tool*, consisting of a palette of functions which then are selected and dragged into the composition editor. A flow engine together with an appropriate middleware then executes this flow model.

During this work an EU initiative was launched, the available outlines showed that research should be undertaken towards Interoperability Development for Enterprise Applications and Software. The aim of INTEROP-NoE (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. In Figure 26 one can see that the inter-enterprise system integration focus is dominant. Nevertheless, the chosen approach of MDA and SOA alignment (Figure 27), 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 allows a certain level of evaluation.

*… The interoperability framework integrates principles of model-driven, service-oriented and adaptive architectures …*

*INTEROP-ESA (2005, Industry Track Session)*

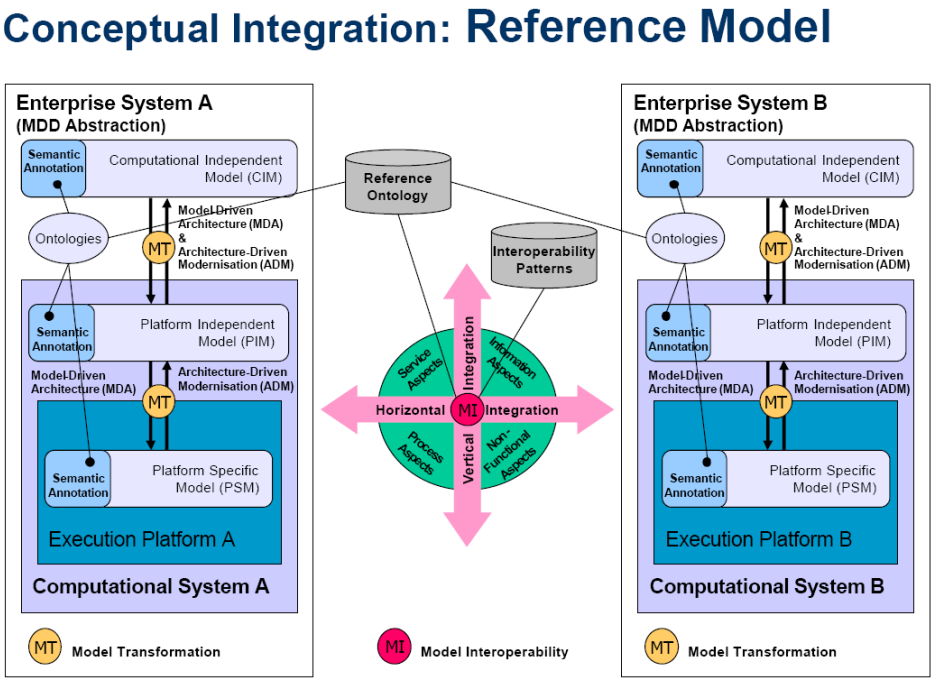


Figure 26. Reference Model of Conceptual Integration (INTEROP D9.1)

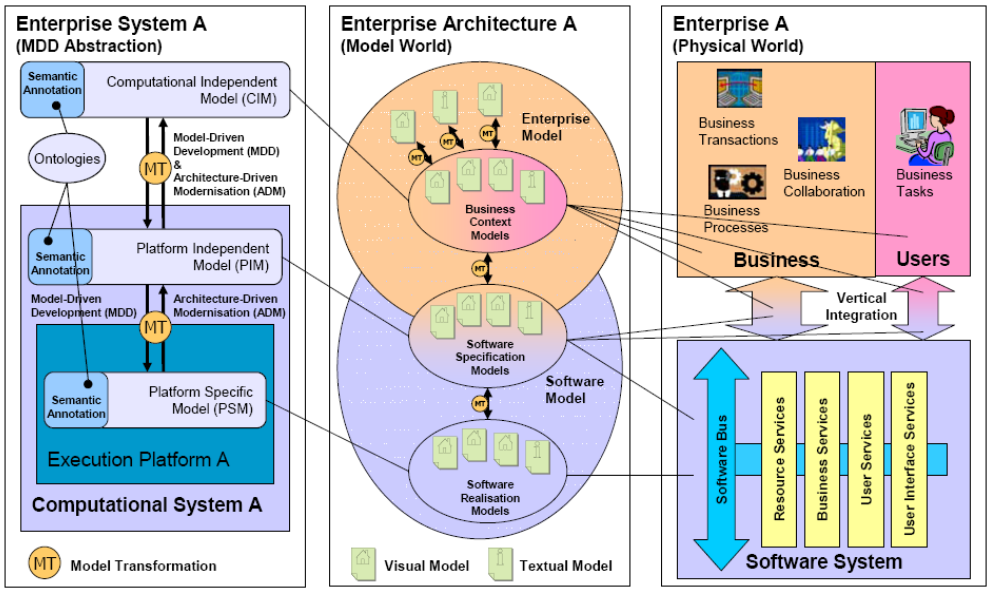


Figure 27. Conceptual, applicative and technical view of an enterprise architecture (INTEROP D9.1)

The second initiative is ATHENA-IP (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications, launched 2004). The deliverables at the time of writing were assiduously examined and the following conclusions were drawn.

INTEROP identifies three main research areas: Enterprise Modelling, Ontology and Architectures & Platforms. After one year a revised work programme enhanced these *Domains* (plus Interoperability Domain) by means of *Task Groups* (INTEROP D4.2, 2004). The *Domains* cover the scientific knowledge base, hence a lot of emphasis was placed on the state-of-the-art knowledge. The available major deliverables (INTEROP D4.1 2004, D6.1 2005, D8.1 2004, D9.1 2004) give a comprehensive overview of the three *Domains* and the result of this joined research is an Interoperability Glossary (INTEROP D10.1, 2005). But they include no new concepts related to the postulated aims. Unfortunately the future oriented Task Groups, which hold such promising titles like Business/IT alignment, IS integration and ontology, did not make any of their results public on their homepage at the time of writing this thesis (March 2006). There was nothing published except for some thoughts concerning interoperability for SME, a proposal so superficial that nothing relevant can be said about the two versions (INTEROP D12.1, 2004 and 2005). Thus, no comparison to the proposal presented in this work is possible, only the recommendation of the excellent research review.

Whereas INTEROP is the nucleus mainly of the university research community, ATHENA sums together some of the big players of the IT industry. Although the useful Reference Models were available from the very beginning, very little relevant information was added to how they could be implemented. Lippe et al. (2005) demand for a 3-level modelling approach (Business, Technical and Executable Processes) and claim that a process abstraction concept is missing. Their survey on modelling languages is worth to be discussed, especially when they claim that UML does not support business context. In such a comparison the UML extension mechanism should be considered. All the more, as UML profiles are claimed to exist (Berre 2005: UML Profile for PIM4SOA; Pondrelli 2005a: UML Profiles for Services, Business Objects and Ontologies) for model driven SOA development. In Pondrelli (2005b) it becomes clear that no new profiles are delivered, but existing proposals (e.g. IBM UML Profile for Software Services) are incorporated in a rudimentary methodology. Berre (2005) presents the results of the ATHENA project after 18 month, the ATHENA Interoperability Framework. It would have been interesting to get more information about ATHENA Service Oriented Interoperability Framework (including Platform Independent Model for SOA (PIM4SOA) & Model Transformations) beyond the description in INTEROP D6.1 (2005) or MPCE Architecture, but the published content is not sufficient for a final conclusion. In addition, the focus on CBP (Cross-organizational Business Processes) with a strong emphasis on MOF (OMG's Meta Object Facility) related model mapping increases the scope which is therefore much broader then the objectives of the single modeling language, intra-organisational approach presented here.

Recently a greater emphasis on Service Oriented Analysis and Design (SOAD) could be observed in the software engineering community. Arsanjani (2004) rediscovers the three model abstraction dimension of reference architecture proposals like CIMOSA when he states that the process of service oriented modeling consists of three phases: identification, specification and realization. But he correctly postulates that it can no longer be an exclusive 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 modeling). For our methodology we adopted the hybrid SOAD modeling approach of Zimmermann et al. (2004), who suggest a combination of OOAD, BPM and EA techniques. It is the aim of this work to enrich and unify these fragments towards a comprehensive SOAD approach with domain specific semantic annotations. Moreover, the methodology has to be validated by means of real-world standards, techniques and applications.

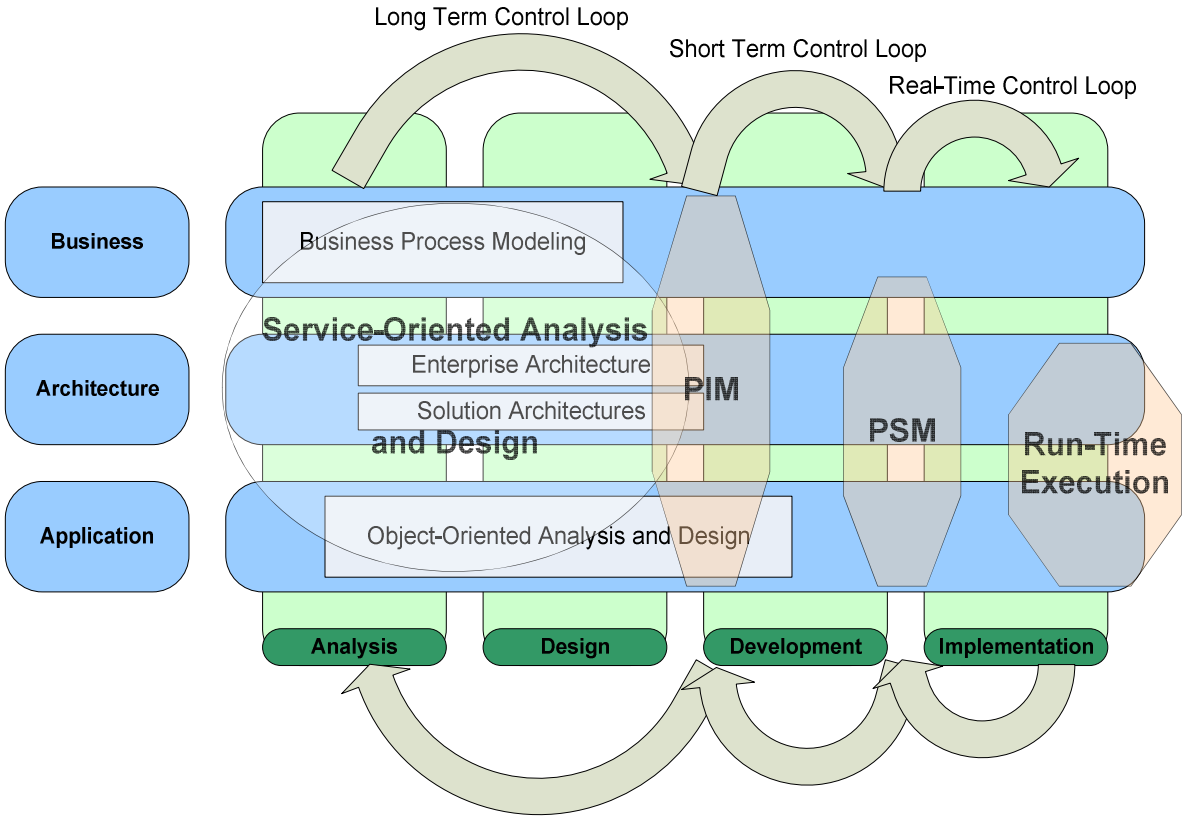


Figure 28. Generic Model Driven Service Architecture approach (adopted from Zimmermann et al.)

## 5.2. Methodology Derivation

The methodological derivation started with the domain dimensions *Business*, *Architecture* and *Application*, each with its own modeling concept. Figure 28 depicts the starting point for the methodological considerations, derived from the guiding principles outlined above. The y-axis represents the domain dimensions Business, Architecture and Application, each with its own Modeling concepts. SOAD has to bring those three together in the life cycle phase *Analysis and Design*. The end result should be a platform independent model (we do not differ between platform and computation independent), which has to be mapped to the actual and potential system assets. Hence it becomes a Platform Specific Model, which will loose some readability for business analysts as implementation details are added. This is the Development phase. The Run-Time Execution is the logical end point, tied to the Architecture and Application domain. This does not mean that run-time data does not generate business relevant key performance indicators. It means that control loops are needed which feed the knowledge gained at the implementation level back to the higher levels of abstraction. Hence such a process is highly dynamic, with interwoven models which translate system behaviour to the kind of notation each system worker can understand. Before we discuss the means which have the potential to glue these phases together, we introduce in more detail our final methodology which is able to support this generic approach.

In Figure 29 we see the resulting methodology specifically for the shop floor domain. To demonstrate the scope of the research undertaken the author has changed the y-axis, which now is in line with the classical hierarchy levels for the shop floor, namely Enterprise Level, Shop Floor Control Level and Shop Floor Field Level (see Figures 2 and 3). In the past the modeling concepts (Business Process Management, Enterprise Architecture and Solution Architectures, OO Analysis and Design) where utilized separately with the emphasis at the corresponding levels as shown in the figure.

First, fast and easy initial modeling 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 modeling 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 on two abstraction levels, platform / computer independent and platform specific. Hence platform specific content can be found in the SFTB as well, for instance in the case of highly standardized industry data exchange protocols like OPC. The complete platform specific information covers 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.

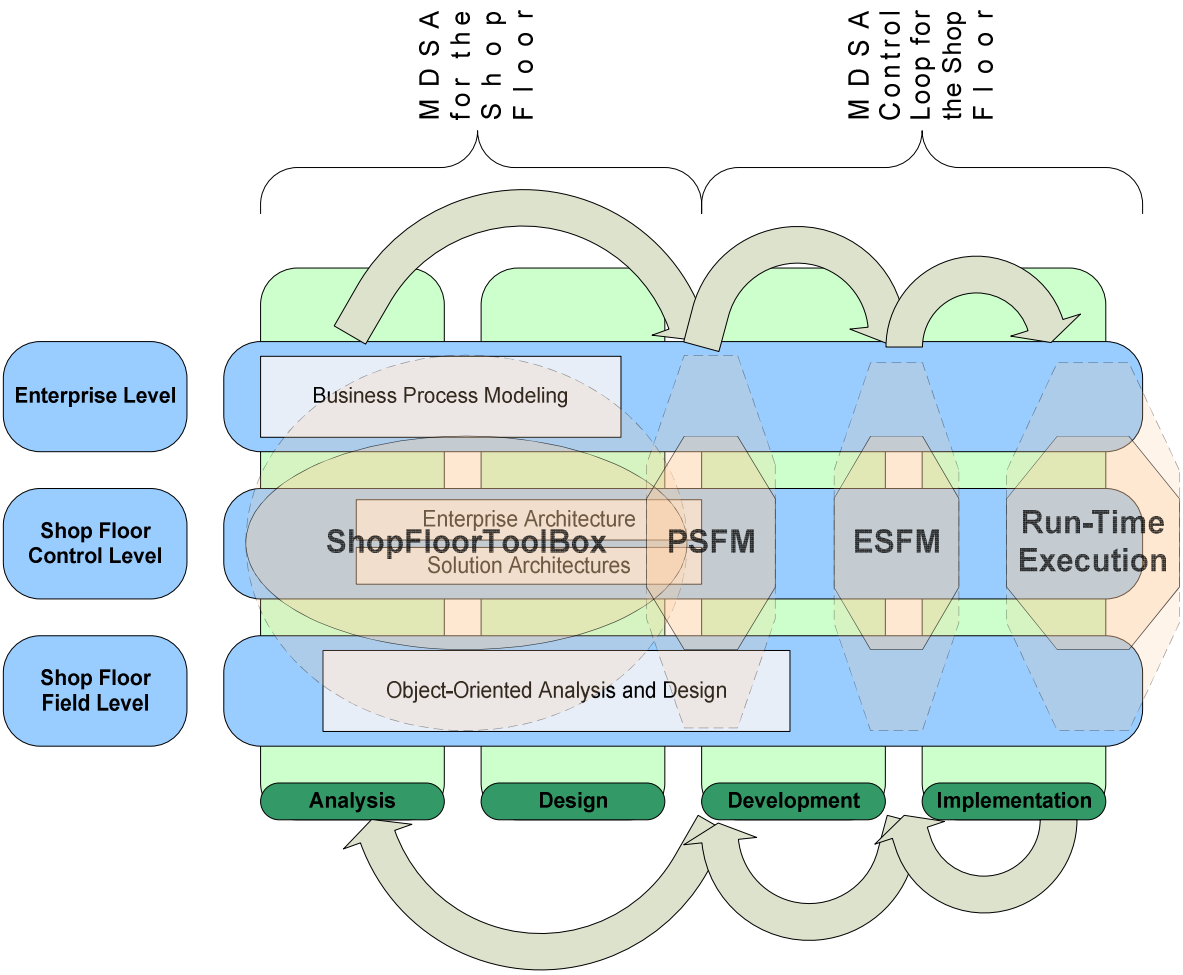


Figure 29. Model Driven Service Architecture for the Shop Floor

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 the 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. The consequences of such an approach regarding BPM explained for business analysts are described in Pfadenhauer and Kittl (2006a).

Crucial for the development of a universal methodology from system models to executable process definitions is the state of the art of the underlying technologies and tools. Thus before the methodology and framework are introduced, the next clause will discuss preparative examinations regarding model driven WS composition. These findings were published and presented at two IEEE conferences (Pfadenhauer 2005c, 2005d).

## 5.3. Model Driven WS-SOA Service Composition

Figure 30 illustrates the three main areas which have to be combined for model driven service composition. Alonso et al. (2004) mention three main elements for WS composition middleware: the *Modeling Environment*, the *Development Environment* (IDE) and the *Run-Time Environment*. The latter one has to execute the coded composition specification. How this specification is achieved, i.e. how and within which environment the necessary specifications and tasks from high level system model down to code execution are fulfilled, is a matter of ongoing discussion. *Tasks* like syntactical and semantic verifications have to be considered, just as service discovery and binding. Which type of *Composition Specification*, the *Composition Model*, the *Model Representation Language* or the *Executable Composition Language*, is best suited for which kind of task, can not be answered definitely yet. The dependencies between these composition specifications on the one hand and their integration within the three environments on the other determine the demand for mapping and testing functionality. The challenging factors depicted in Figure 30 establish our collection of criteria for evaluation of existing approaches.

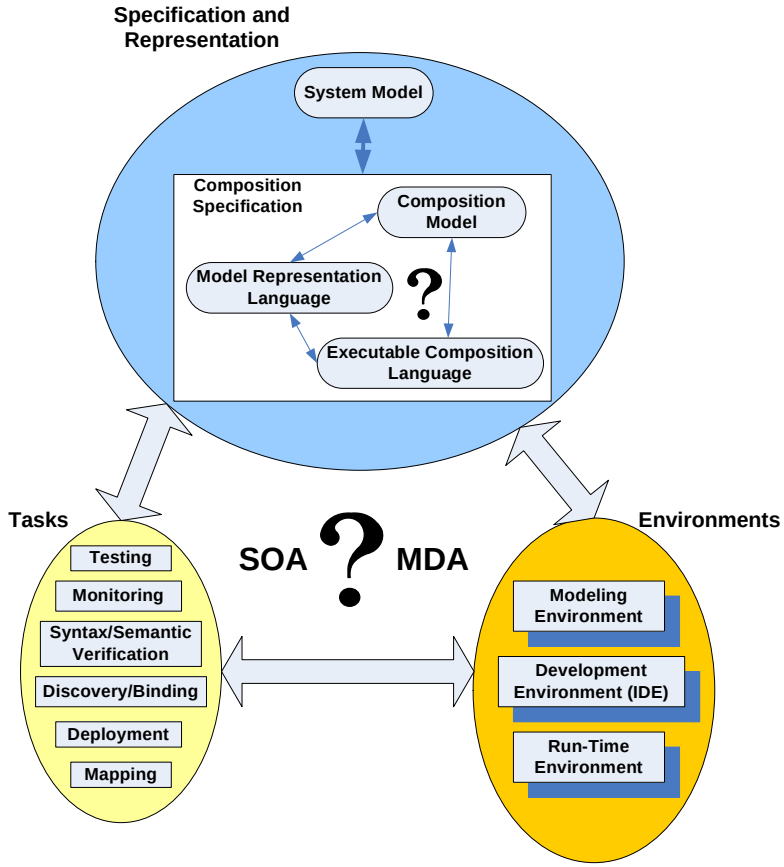


Figure 30. Composition challenges regarding model driven WS architectures

The number of enclosing real-world approaches especially for top-down model driven WS composition with an appropriate tool support is limited.

The reasons, therefore, are amongst others:

- Still a number of process mark-up languages, the "missing link" between model and execution, like BPEL4WS or WSCI/BPML exist. None of them can be called a widespread implemented standard yet.
- Model and code have to equally support certain control patterns, the evaluation of this is still in progress (see Dumas & ter Hofstede 2001, van der Aalst 2000). It is likely that these patterns have to be further expanded by means of the emergence of second generation WS technologies in conjunction with the service oriented architecture (Dustdar 2004, Erl 2004).
- Model mapping and transformation is still limited to static entities like classes and components, the complex task of dynamic information transcription is just at the beginning.
- The transformations have to be completely bi-directional, a requirement which is hard to achieve.
- Intelligent validation and testing mechanisms between the mapping steps hardly exist yet.

To sum it up, the proceeding from WS Composition graphs to executable service flows is in the state of lively theoretical discussion, although first implementations already exist. At this time two rather distinctive approaches can be observed, one with model environment focus, the other with IDE focus. They also stand for completely different ways how the model driven concept can be interpreted. Due to the need for publicly available tools we focused on an IBM and a Microsoft scenario.

### 5.3.1. Top-Down Model Driven WS Composition

Generally speaking, the first approach illustrated in Figure 31 postulates the concept of model enrichment through extensions combined with elaborate transformation and mapping mechanisms, most prominently represented by means of OMG's MDA (Model Driven Architecture) (see above). This ambitious concept demands a strict separation of concerns, expressed through a layer stack consisting of different levels of abstraction. To achieve this, the models have to cover a broad set of requirements.

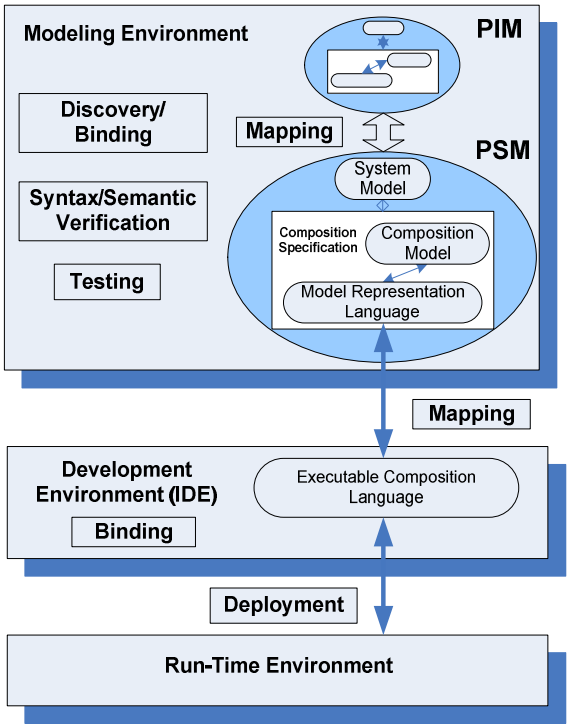


Figure 31.Top-down model driven WS composition

As an example for such sophisticated requirements, the complex task of PIM (Platform Independent Model) to PSM (Platform Specific Model) mapping shall be mentioned. This transformation step can be extended by an additional executable UML layer, introducing mark-ups and languages like ASL (Action Specific Language). This intermediate step makes the model executable, especially valuable for testing before mapping.

Skogan, Grønmo and Solheim (2004, Grønmo and Solheim 2004) present a very interesting approach. They introduce not only a technique how to import WSDL service descriptions into UML, but also how to use UML activity graphs enriched by means of certain UML extensions to define WS compositions. In addition they use XMI in combination with corresponding XSLTs to achieve a transformation in several execution languages, at the moment BPEL4WS and WorkSCo are supported.

The use of templates and patterns can be applied, either for activity graph design (Förster 2002), or for consistency analysis.

Voigt (2003) has implemented the latter by means of UML-CSP (Communicating Sequential Processing) language mapping. The benefit of this high level top-down approach is a stringent process from abstract domain models down to executable code.

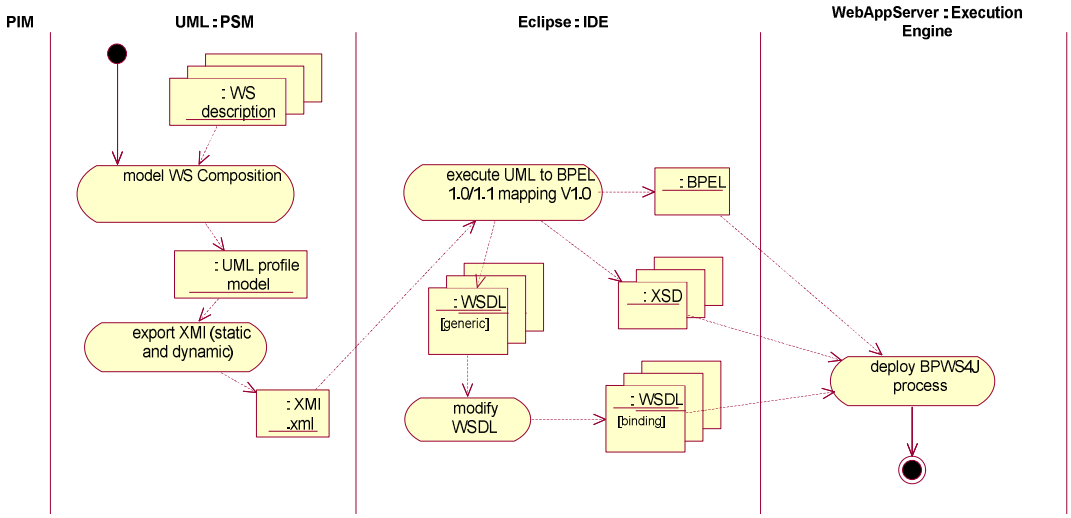


Figure 32.IBM UML-BPEL-BPWS4J proposal

The IBM approach (see also Mantell 2003) depicted in Figure 32 begins with an UML model, which is specific for the BPEL4WS platform through extensions following the "UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0" (Amsden et al. 2003). Modeling according to the profile makes the WS descriptions (port types, operations, messages) of the involved services necessary, but no WSDL import support is available. Hence, this information has to be extracted and included into the class diagrams (static view of data types / messages, protocols and roles) and activity diagram (dynamic view) by hand.

After finishing the model gets exported into XMI format, which then is imported into an Eclipse 2.1.3 IDE java project. There an add-in performs the mapping to BPEL4WS 1.0 or 1.1, generating the WSDL, XSD and BEPL files. Bindings, location paths and service links are added by means of a WSDL modify tool.

Deployment is separated from the IDE, afterwards the BPEL process can be executed via the BPWS4J 2.0 runtime engine.

### 5.3.2. Bottom-Up Model Driven WS Composition

The second approach (Figure 33) is best described as bottom-up, because not the portable architecture model is the starting point, but the existing process development and execution possibilities within a certain vendor environment. The vendor independent visualization is just an extension of the central graphical drag & drop interactions offered by the IDE. In addition to platform specific (PSM), the IDE models have to be called vendor specific (VSM).

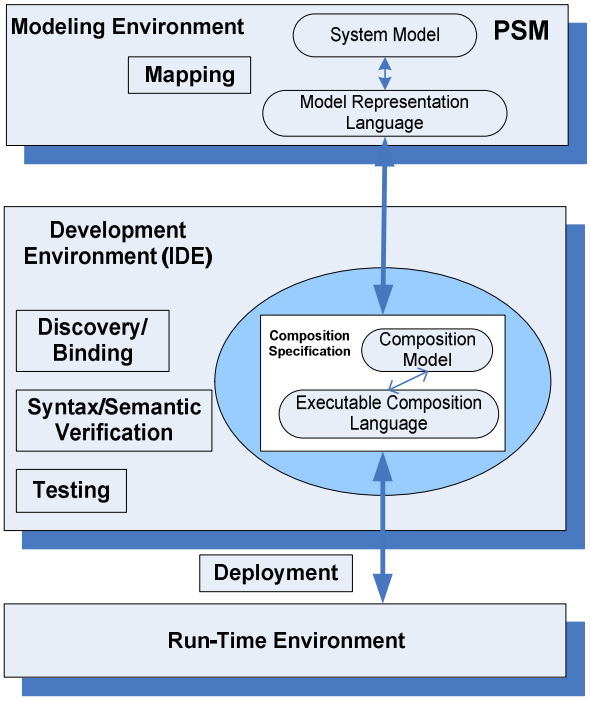


Figure 33. Bottom-up model driven WS composition

The Microsoft proposal (Figure 34) can be described as follows. In the Visio Orchestration Designer a proprietary modeling semantic is used to design the basic flow pattern, allowing fork, join, group, decide and loop constructions. An add-in exports this activity graph into a BizTalk Orchestration XLANG/s .odx format, which can be imported into the BizTalk Orchestration Designer, which again runs within the Visual Studio IDE. The rudimentary flow is now supplemented with ports and messages, which are created by means of wizards utilizing the port types and message types retrieved from the added web references. All WSDL, XSD and the .odx files are generated and updated with strong graphical support. The process can be deployed within the IDE using an administration console.

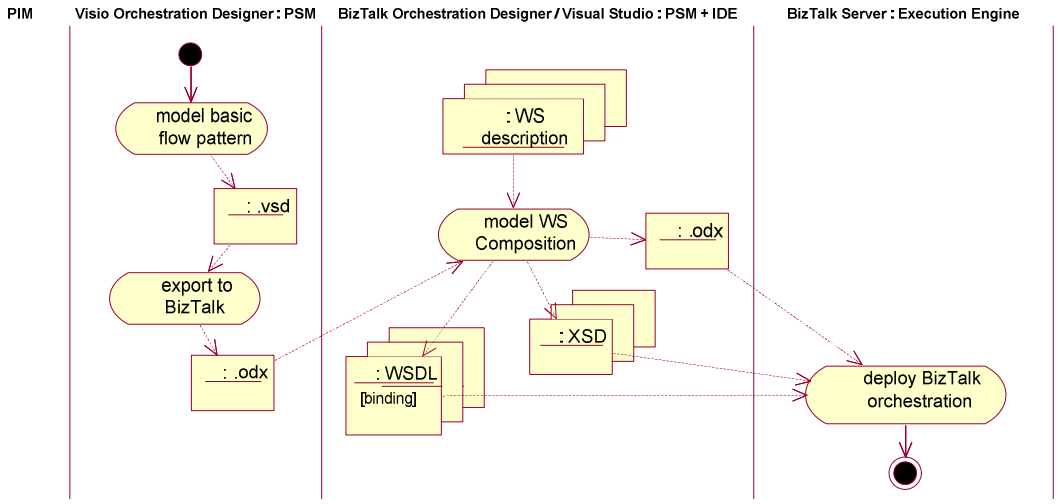


Figure 34. MS Visio-XLANG/s-BizTalk Server proposal

## 5.3.3. Comparison and Evaluation

As we mentioned above, both approaches are model driven Web Service orchestration implementations, but with very diverse emphases. In chapter two we emphasized current challenges on WS composition. This discussion led to a collection of criteria which in the following shall be used for comparison purposes (Table 2). The test environment was the following:

Table 1. Top-down and bottom-up comparison test environment

| | IBM – Top-Down Approach | Microsoft –Bottom-Up Approach |
|---|---|---|
| Modeling Environment | Rational Rose Modeler Edition 2003 + Rose XML Tools 1.3 plug-in | Microsoft Visio 2003 + Orchestration Designer for Business Analysts |
| Development Environment | Eclipse 2.1 + BPEL4WS Editor 2.1 plug-in | BizTalk Server 2004/Visual Studio 2003 |
| Run-Time Environment/OS | BPWS4J 2.1 + WebSphere Application Server-Express 5.1 / Windows XP pro SP2 | BizTalk Server 2004 / Windows Server 2003 |

Generally speaking, IBM tries to design the WS composition at a higher level of abstraction, using an UML profile for nearly complete process definition. Therefore, the mapping effort is rather high; unfortunately no syntax validation takes place before XMI export, which makes the failure risk during the subsequent BPEL4WS mapping very high. The IBM approach offers at no level WS discovery or reference mechanisms.

Table 2. Evaluation results

| | IBM "top-down" | MS "bottom-up" |
|---|---|---|
| **Environment** | | |
| Modeling | substantial | rudimentary |
| Development | code oriented | graphical/wizards |
| Run-time | BPWS4J | BizTalk Server |
| **Specification** | | |
| System model | no | no |
| Composition model | enriched UML | poor + proprietary |
| Model representation language | standardized XMI | proprietary |
| Executable composition language | standardized BPEL4WS v1.0/v1.1 | proprietary XLANG/s |
| **Tasks** | | |
| Testing WS links | no | within IDE |
| Testing orchestration | run-time | run-time |
| Monitoring | poor | strong |
| Syntax Verification | limited | strong |
| Semantic Verification | no | no |
| Discovery | not supported | search functionality |
| Binding | static | static |
| Deployment | isolated | integrated |
| Mapping | isolated | integrated |

The manual WS description import, i.e. the WSDL file transformation into different types of stereotyped classes within UML is tedious. On the other hand the XMI format containing the whole business logic and platform specific information offers theoretically a certain flexibility concerning model reuse and transformation. In practice the re-import of created XMI files into Rational Rose, our UML tool, was not complete, manual adjustments were required. The necessary support within the IDE is reduced to WSDL modifications. The XMI mapping to BPEL4WS itself takes place automatically, but again without XMI validation. If an error occurs, the mapping stops without error handling. A BPWS4J Editor plug-in for the Eclipse 2.1.3 IDE offers enhanced verification functionality and more important rudimentary graphical support for process editing. Nevertheless, the hierarchical visualization is much less expressive and functional than the one within the MS Orchestration Designer. Also concerning deployment, monitoring and testing the MS approach is superior, but that is what we expected. It should be mentioned here the pointlessness of a comprehensive IDE comparison. In that case IBM WebSphere Studio rather than Eclipse would have to be the challenger, resulting in a comparison between two IDE centric approaches.

The second proposal places the Web Service orchestration at a lower level of abstraction, within the IDE and, therefore, establishing a tight coupling between visualization and coding. The initial Visio modeling in the MS proposal offers very limited additional value, since beside the control flow no additional information can be included. Moreover, for control flow modeling the IDE functionality is as easy to use. The diagram is not integrated within other stencils. Therefore, the integration in an architecture model is missing. After the import into the IDE the model is platform specific twofold (WS/XML and .NET). Once the basic service flow is set up, the

service binding is very comfortable by means of drag & drop and wizard functionality. At this level code and graph are updated real time, a benefit which is only possible by means of a tight integration into the IDE. Although the underlying XLANG/s language is proprietary, restricted BPEL4WS 1.1 import and export functionality is offered, which allows for a certain degree of portability.

Both approaches are concerned with single process definition and do not take into account a system model from which the process may be derived. That is to say, the UML allows for further extensions referring to this issue, whereas the Visio model does not offer this flexibility. As one can see in Figure 30, the dependencies between a single composition specification and an overall architectural model are in our opinion a vital part of a methodology which supports life cycle oriented service composition. Process lifecycle management without a complete model covering all relevant interaction aspects is not possible. Reactive semantic verification mechanisms should be implemented twofold, firstly concerning the control logic (e.g. the detection of deadlocks) and secondly concerning inconsistencies regarding the semantic at different levels of abstraction after manual intervention. Both ignore the first requirement, and only the MS approach inhibits inconsistencies between code and graphical representation due to the tight integration within the IDE (but not between the Visio model and the IDE graph). After our discussion it should be evident that for single process definition one can choose between a top-down and a bottom-up approach, but for service oriented architectures the top-down approach is crucial.

## 5.4. Lessons Learned for the Proposed Methodology

Above two general approaches of process design and execution have been compared, one with strong focus on the abstract, platform and vendor independent model, which is semantically rich enough so that an IDE is exclusively needed for (complex) mapping tasks. The other with the focus on “applied” modeling within the IDE, supported by a rudimentary and abstract graph describing the control flow. The latter bottom-up approach, represented by MS BitzTalk Server, has its main advantage concerning easy integration of existing Web Service definitions, which is not possible within the top-down approach. On the other hand the integration of the Visio or the Biz Talk Orchestration models in an overall, syntactical homogeneous architecture model is not possible, thus these directed graphs have to be built from the scratch. Here the UML-BPEL4WS approach offers much more possibilities of integration in an enclosing MDA concept.

In the following, challenges for future model driven process engineering proposals must be discussed. We demand for a computation independent system model as the starting point, which supports not only isolated process design, but architecture modeling. The behavioral views within these models have to be the basis for initial Web Service composition design.

It is then necessary to investigate, whether the presented approach is suited for shop floor domain usage. Therefore, some core processes of production operation management are implemented in a use case scenario, allowing for tests regarding applicable mechanisms for process model and process specification synchronization. Still another open question is how the process models have to be embedded in the overall architecture model, i.e. how operational patterns, which synchronize between static architecture structure (*the sum of all possible processes*) and single process lifecycle management, can be established.

### 5.4.1. Computation and Platform Independent Modeling

Following the MDA notation, IBM's UML-BPEL proposal starts with a UML 1.4 compliant definition of a PSM (the platform is WS/XML) and transforms (via separated mapping rules) it into a model specification language by means of XMI, a standardized XML language. The core MDA concepts, different levels of abstraction, separation of concerns and the model transformation paradigm are, therefore, fulfilled. This is not true for the MS proposal, where both the initial and the BizTalk Orchestration model are proprietary and not transformable into a standardized format like XMI. Both approaches do not take into account the mappings from computation or even platform independent process descriptions, which is in our opinion their main weakness regarding MDA principles. Enterprises will not make the same mistake twice like it happened in connection with proprietary ERP (Enterprise Resource Planning) systems in the past. Often the business logic was modelled and defined tightly coupled to a certain platform. The WS/XML platform will be one integration technology among others in service oriented architectures, thus platform independent modeling is crucial.

### 5.4.2. Integration of Modeling Techniques

It is also a fact that concerning process modeling the use of a unique syntax is last but not least within the area of WS orchestration far from becoming true. Historically, two more or less separated paradigms have evolved, the business and the information system view, resulting in two groups of modeling techniques, one for BPM (Business Process Modeling) containing IDEF0, Petri Nets, EPC (Event-Driven Process Chain), Flowcharting etc., the other for ISM (Information System Modeling) containing Data flow diagramming, ER diagramming or UML. Integrated design strategies rarely have been put into practice (Giaglis 2001). With the emergence of Web Service-based process design and execution an alignment of these two views is vital for successful BPM (Business Process Management) more than ever. Therefore, and because of the reuse of process repositories already existing within the organizations, the integration possibilities of proprietary business process models within the above mentioned techniques should play a more important role. In the context of a top-down approach, mappings from XML representations of widespread used and well-defined modeling

techniques (EPC Markup Language in the case of EPC, or PNML Petri Net Markup Language in the case of Petri Nets) to platform independent MDA-UML models seem promising. OMG's Business Process Definition Metamodel (OMG 2003), a UML 2.0 profile, aims at this goal. This proposal supports the mapping to a common metamodel and thus facilitates the communication among a variety of process models. We mention in this context ArcStyler, which is in the first place a classic MDA tool vendor for software engineering purposes, but with the MDA-Business Transformer for ARIS (Interactive Objects Software 2002), an eEPC (extended Event-Driven Process Chain) to UML mapper, they offer an interesting approach of business and IT view alignment. Unfortunately they do no support UML to BPEL4WS mapping. Van der Aalst (1999) has successfully undertaken a similar task, that is to say a mapping between EPC and Petri Nets. Modeling techniques alignment can also be achieved at a lower level of abstraction, neglecting platform independent representation through direct composition language mappings. This would be a similar, platform specific solution like the UML-BPEL4WS example from IBM described above. For a bottom-up approach, mappings to proprietary modeling syntaxes, embodied e.g. in MS BizTalk Orchestration Designer or IBM WebSphere Business Integration Modeler, would be needed.

### 5.4.3. System Modeling

It was already emphasized that the dependencies between a single process model and an overall architectural model are not considered within the two approaches. To see the process isolated from its environment has many disadvantages:

- Inter process dependencies like synchronization or process hierarchies (nested sub-processes) are missing. Hence, appropriate business rules and constraints have to be coded. Monitoring by model is no longer possible.
- Service repositories are missing at initial modeling level. The design of the activities takes place without knowledge about their availability according to necessary QoSs (Quality of Service).
- Within architectures different paths exist and new evolve to achieve a certain process aim. Without an architecture model and the knowledge about the given connectivity, one has to design a single process model for every control flow, not knowing whether the needed interoperability exists or not.
- The consequences of a change in the architecture or the introduction of new service providers for the process designs can not be monitored accordingly.

Ragarding the MDA approach, OMG's Business Process Definition Metamodel (OMG 2003) again seems noteworthy, because the final specification is expected to achieve a metamodel that complements existing UML metamodels so that business processes specifications can be part of complete system specifications to assure consistency and completeness.

### 5.4.4. Platform Specific Model Completeness

Recent second generation Web Service Technologies like WS-Reliable Messaging or WS-Policy (Erl 2004) are not embodied within the platform specific modeling environments, although they are fundamental for transaction and context implementation regarding service-oriented inter-organizational integration (Dustdar 2004). Their absence in both modeling approaches means these concepts have to be included belatedly, an advancement that jeopardizes model and implementation congruency. In our opinion a process lifecycle management without a complete model covering all relevant interaction aspects is not possible.

### 5.4.5. IBM and Microsoft Next Generation Tools

IBM's next generation of integrated modeling tools focuses on the Rational Software Modeler / Architect product suite which will integrate business modeling with UML modeling. It will then be possible to import and export WebSphere Business Integration Modeler projects as UML 2.0 models to and from Rational Software Modeler. It seems as if this leads toward an integrated approach, where top-down and bottom-up strategies can be chosen scenario dependant. Microsoft seems to stick to the bottom-up approach, with emphasis on code / design synchronization and validation. In addition, the Visual Studio 2005 Whitehorse project, a group of graphical designers that support the design and deployment of Web Service-based systems, will foster the reliance on WS platforms.

# 6. MDSA Methodology Description

This methodology backs the core project objective concerning enterprise architecture. It is the opinion of the author that the present vertical and horizontal enterprise organisation will soon be obsolete. In the context of SOA the repository of services (with a description of semantics) is at the core, answering the questions how the connection to the services can be established and what message structure is needed for communication. Some major developments which strengthen this believe were already mentioned when the motivation for this project was discussed. For a highly distributed system like SOA a model based methodology has to exist which supports design time as well as run time decision making. With the Model Driven Service Architecture (MDSA) we propose an approach which fulfils major goals:

- System functionality focus: According to the basic SOA paradigm a service repository is crucial. The MDSA suggests a service/process repository as the core modeling concept incorporating the physical, deployed repository. At all abstraction levels it must be clear what functionality is available and how connectivity (data objects and interaction channels) can be achieved.
- Abstraction level focus: Much discussion takes place about flat hierarchies, and the SOA is seen as an enabler to overcome any hierarchy at all. We believe that due to the need for different levels of abstraction another kind of hierarchy has to be established. Not a hierarchy of enterprise organisation or IS, but a hierarchy of different levels of abstraction of services. The MDSA supports the organisation and alignment of different abstraction levels.
- Life-cycle focus: The MDSA approach is not just a guideline for single SOAD projects, but aims especially at the support of the system architect during the whole system life-cycle. Therefore dynamic system behavior under uncertainty is an important issue, resulting in the concept of closed control loops for long term as well as short term decision making.
- State-of-the-art technology focus: MDSA at the highest abstraction level is a methodology for distributed systems design by means of long term and short term control loops. To proof the assumptions an implementation is necessary. First the implementation of the MDSA methodology in a modeling environment, resulting in a UML based approach. Second the implementation of a demo scenario to demonstrate the application of the MDSA during the complete system life-cycle. For this the Web Service architecture shall prevail.

In the following we do not start with the analysis and design phase of Figure 29, but with the development and implementation phase. This makes sense because it will outline the basic requirements concerning abstraction levels and service/process repository in the analysis and design phase.

After the introduction of the Shop Floor Tool-Box concept for the latter, we have to turn to the question what techniques and technologies are best suited for information system engineering following the MDSA methodology.

## 6.1. MDSA – Development and Implementation

The platform independent Particular Shop Floor Model (PSFM, Figure 29) 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 platform specific Executable Shop Floor Model (ESFM) concerning model refinement. The latter serves at a tactical level for the (re)design of service flow definitions which are semantically rich enough for executable code generation. The dynamic views of both levels together constitute the basic service/process repository. This model views have to be kept consistent with the deployed SOA repository. It provides patterns specifying the particular system processes, each on the one hand in a more generic (computation independent) model for the use within the PIM and on the other hand particular process models for the interaction within the PSM. Alternative service flows, different versions or diverse technological specifications of one and the same process are, therefore, separated from the actual valid models and enlarge the service repository. Templates allow for fast process creation.

This proceeding takes into consideration the necessity of business- and IT-view alignment, allowing e.g. architectural views or process definitions at distinct levels of abstraction. The purpose of MDA is a bi-directional, consistent procedure from system modeling (domain modeling) to application building and connecting. To handle this goal in the daily enterprise business, a clear methodology has to exist to control and adjust system behavior. In system theory therefore the model of control loops was elaborated for detection of abnormality between *to-be* and *as-is* behavior. In many domains this concept of system control was introduced. In the context of MDA and SOA the implementation as *Management Control Loop* for enterprise control is especially interesting, because there the level of decision making (strategic – tactical – operational) is considered through a control loop cascade. A transformation of this concept leads toward a *Model Driven Service Architecture Control Loop* as can be seen in Figure 35.

The PIM at the strategic level consists in our approach of the infrastructure in general, which again consists of services provided by distributed entities and the communication infrastructure containing hardware and data entities. In this part of the model various process definitions can be found, it represents the *sum of all possible processes*.

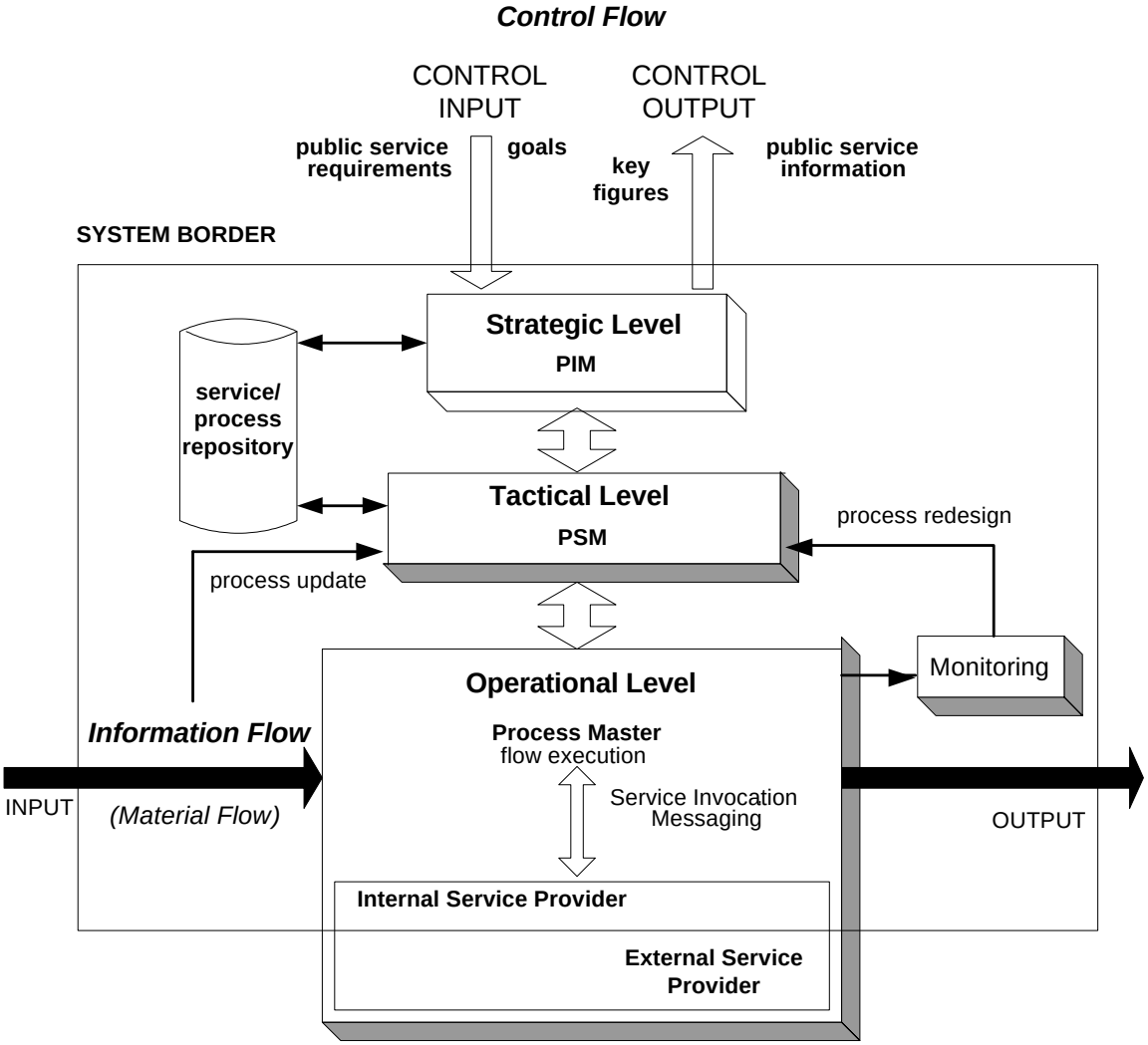


Figure 35. Model Driven Service Architecture Control Loop

Thus we want to provide a service/process repository which contains standard processes within the system, in our case the shop floor. This process repository can be used for PIM *as-is* evaluation and therefore PIM *to-be* creation. Far more important it is in the tactical level, where the executable process definition takes place and where process storage is absolutely needed. Process update and redesign at the tactical level utilizes the process repository. This means that according to the actual information flow or the monitoring results process definitions can be switched between actual valid PSM and repository.

The figure shows that all structural model changes (e.g. new service providers), whether initiated through feed forward/feed back control flow or through system environment control flow, have first to take place in the strategic level PIM and than are mapped to the tactical level PSM due to consistency. Explaining the difference between PIM and PSM in this context we refer to the work of Leymann et al. (2002), who introduce the so-called $W^3$ space to describe the execution of a flow instance in a flow model. A control flow consists of a chain of points in a three-dimensional space with the dimensions *what* (activity), *who* (responsible role) and *with* (tool for activity

processing). In a PIM *what* and *who* dimensions are fixed, but the *with* dimension remains at a high level of abstraction, not considering detailed specification for instant execution. The latter one is added through the PSM mapping, when platform specifics are included in the flow definition. Hence in the PIM it is sufficient to model the data flow coarse grained with focus on the semantic rather then the syntax. When we talk about platforms we have a broader understanding of this term compared to the original MDA definition, not just IT-infrastructure like middleware solutions, but also shop floor specific communication standards and non-IT tools like writing down information. The PSM with the platform specific description of the connectivity and distributed functionality within the system, which represents the *sum of all executable processes*, is the basis for the tactical level where the final process definition takes place, interacting with the process / service repository.

As was mentioned in the SOA introduction, a system seldom stands alone but is part of a parent-system and therefore service hierarchy. The external service requestors are not interested in detailed information about internal model changes as far as they do not influence existing public service behavior. On the other side public service requirements, in addition to system goals are the input parameters of the external control flow. Key figures for the management of the system, like lead times, process execution errors etc. have to be defined to control overall system performance.

## 6.2. MDSA - Analysis and Design

A methodology from generic constructs and templates to the PSFM has to be delivered.

Fast and easy initial, computation independent modeling 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 high level model construct collection called *Shop Floor Tool-Box*. By means of scenario specific selection of generic artefacts from the toolbox, a sophisticated PSFM evolves. Residing at a strategic level this model does not include detailed information enabling automated flow execution, but takes into account the socio-technical structure of the domain. Exceptions are self-contained SFTB packages which describe highly standardized technologies or complete applications with a given implementation. Thus, after the analysis and design phase, certain processes and therefore services can exist in a platform specific occurrence as well.

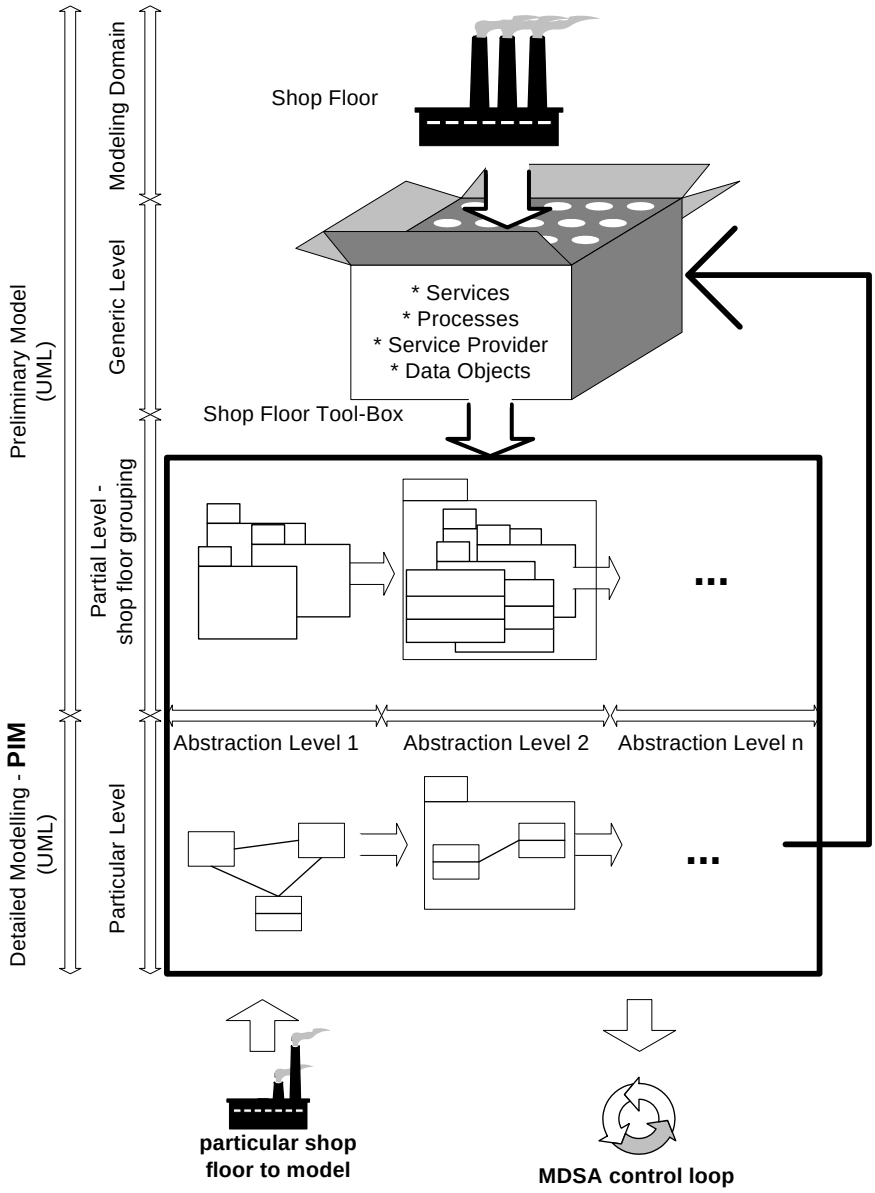


Figure 36.The concept of a Model Driven Service Architecture for the Shop Floor (structural view).

## 6.2.1. Shop-Floor Tool Box

Above we described a methodology that helps to manage the system control in the enterprise with the PIM and the service/process repository for this particular system as the main input. The PIM differs of course for every given system. This is especially true for the shop floor, where a wide range of system architectures can be found. In contrast the process repository is rather steady, because the necessary services and their logical invocation order to achieve the process goal (*what* dimension) are similar within different shop floor environments. What strongly varies are the *who* (either the worker or the NC-machine is responsible for tool setting data request) and the *with* (either the request gets transmitted by hand or Ethernet) dimension. Thus the project

aims at the methodology and tool creation which eases the modeling of platform independent models for service oriented architectures in the shop floor on the one hand and a standard service / process repository on the other.

Therefore, a high level model of the shop floor is needed which contains a hierarchically ordered collection of services, necessary to achieve typical, nested process functionality within different shop floor environments. We call this high level model *Shop Floor Tool-Box*, because it is a tool which allows the fast visualization of processes and services in a PIM within a given shop floor architecture. The tool consists of an activity model (*what* dimension), service provider (*who* dimension) and data entities (together with the service provider functionality constituting the *with* dimension). The activity model of the tool represents the control and information flow descriptions (behavioral view).

The *Shop Floor Tool-Box* allows for structural and behavioral modeling with standardized and encapsulated objects and packages of the shop floor, which offer services at different levels of abstraction. Additionally, the tool box considers standard processes and their varieties within the shop floor (order fulfilling, tool management, etc.) without fixing the responsibilities of the control objects for specific activities (this will be done during the analysis and design of packages in the partial level and refined in the particular level PIM creation of a given system). Thus the first phase of the project was occupied with the collection of services, control objects and entity objects likely to be found in the shop floor, hierarchically ordered using packages and composition. The services needed in the shop floor are separated and not allocated to the control objects, but are linked to rudimental standard processes in the process repository. During the work for the SFTB it soon became clear that such a tool needs a stringent and logical framework for model organization. The usability within the modeling tool turned out to be critical as well. After some not really satisfactory proposals the ANSI / ISA organization published part 3 of the ANSI / ISA 95 standard which turned out to fulfill most of the requirements defined for the SFTB framework. Hence the author decided to organize the SFTB framework according to this standard.

Because selected groups of shop floors and their particular resources, e.g. Flexible Manufacturing System versus workbench production, show typical constellations of service – control object – data object conjunction, we introduce a partial level between the generic level (toolbox) and the particular level (particular PIM / process repository), according to the CIMOSA Modeling Framework. In contrast to the latter, we do not create complete reference models, thus the partial level contains only groups of process implementation (sequence and / or activity) diagrams.

At the particular level a specific initial PIM / process repository is the aim, taking a proper partial level shop floor group model according to the given shop floor environment and adjusting it until it is accurate enough as a starting point for creation of a MDSA control loop.

#### 6.2.1.1. Modeling Tool-Box purpose

The *Shop Floor Tool-Box* serves two purposes. First it has to be flexible enough to support fast and precise modeling of any given shop floor architecture in the context of service availability and distribution, which is important for system analysis, reengineering and requirement definition for system extension. These models allow us on the one hand to get an overview about the current configuration and to show the improvement potential, independent from the planned architecture.

On the other hand the aim is a SOA, thus the model syntax and semantics supports the required constructions. This has to be considered when the SFTB is filled with the structural and behavioral building blocks. Compared to similar approaches like that of Jin et al. (1998), who present a *Short Period Modeling Method* in the preliminary modeling phase for the user's requirements capture and analysis, we do not want to use different modeling techniques for different process phases. Still, we follow the above approach with regards to the principle aim, providing the tools for fast and accurate requirements gathering, analysis and further, more detailed modeling. The use of UML, the state-of-the-art modeling technique for software engineering, will avoid a gap between the preliminary modeling phase and the detailed modeling phase, which will be strongly influenced by IT concerns in the context of process automation. This SOA oriented methodology is possible because there are standard activities in the shop floor. With a given functionality the outputs and inputs of the sub-systems realizing the activities stay the same. The only difference is the behavior regarding the operation allocation between the constructs, not the operations themselves.

## 6.3. MDSA – Assumptions, Technologies and Tools

Although our methodology is SOA implementation independent, for the demo scenario we had to consider the given restrictions just like in any other real-world project. First, we wanted to include some Web Service based HMI applications for NC-machine control, called *Cell Integrator*. Secondly, we have discussed the increasing importance of *Internet Based Manufacturing*, thus applications and devices enabled for internet technologies should be preferred. In line with the fact that WS-SOA is the most standardized and elaborated SOA technology stack we decided to make it the main implementation platform. Human interaction is the second focus. The aim was to put human interaction tasks on an equal footing with fully automated processes. This is on the one hand a functional necessity, because a lot of control tasks proofed to be unsuited for full automation (e.g. detailed production scheduling). Hence human knowledge is indispensable. In modern production management this fact is strongly emphasized in terms of high-skilled workers for high-tech equipment like NC-machines or when it comes to continuous process improvement projects. Thus the scope for decision-making increased for shop floor personnel, but the basic information push mechanism did not change. A SOA gives the worker the freedom to

pull the information actually needed and not the one which was thought to be essential at process design time. Therefore, SOA applications which support messaging and broadcasting subscription in UDDI have been designed. On the other hand such a scenario leads to much broader architectural requirements, asynchronous communication patterns for example are a must for personnel notification. Or time constraints in request-response interactions are different if one of the parties involved is dependant on human activities.

We already mentioned the different approaches regarding SOAD. The author strongly believes that only a combined approach can be successful, matching the given asset structure measured against high-level requirement business models. Thus a comparison of top-down and bottom-up approaches for model driven WS-composition was conducted, 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 a combined *Specification and Representation* approach depicted in Figure 37 was implemented, with an UML 2.0 based methodology for high-level use-case and business scenario modeling (system model) resulting in platform independent service collaboration views which utilize a modified IBM UML 2.0 Profile for Software Services (Johnston 2005)). Johnston's paper is not free of contradictions. The stereotype *Service Partition* has a constraint which states that any owned part shall be a class stereotyped *Service Provider*. This is the opinion of the author as well, but the semantic description restricts *Services* or other *Service Partitions* as nested parts. Secondly, the *Service Collaboration* stereotype participants are restricted to *Service Providers* in the text, but the class diagram shows that the constraint regards *Services*. To make matters worse, the Rational Software Modeler profile plug-in example uses *Service Specifications* to represent the collaboration of other services. The latter case is the one adopted in the modified profile.

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 1 and the *Activity Models* of Part 3) and the IBM Profile for Business Modeling (Johnston 2004, Ng 2002, Rational 2001). Corresponding templates ease the model management.

Other proposals for UML 2.0 business modeling profiles (Sinogas (2001) or the related one by List and Korherr (2005)) partly adopt and as far as the author could evaluate can not outperform the latest release of the Rational approach. Although the latest one, based on UML 2.0 claims to do so.

List and Korherr (2005) present two perspectives, a *Business Perspective* (static view) and a *Sequence Perspective* (behavior view). The general stereotype *Business Process* extends the *Class* classifier in the UML 2.0 meta-model. But this makes the model organization more difficult during process decomposition than using the *Collaboration* classifier. While the differentiation between *Core-*, *Support-* and *Management Processes* and their relation to *Internal-* and *External Customers* in the *Business Perspective* is in reality hard to implement, it is also questionable that the meta-model for the *Sequence*

*Perspective* can be chosen completely situation dependant. Thus, unfortunately, the important link between the two perspectives remains unclear, although it is stated that one of the contributions of this proposal is that "The UML 2 profile for BPM can be easily extended and mapped to Business Process Execution Languages (BPEL)." The *Detailed Process Diagram* defined for the business perspective, mentioned as a "… link to the sequence perspective." is not explained any further. So we do not know how the software developer, for whom the profile represents the business context and business requirements, will integrate the new abstraction level into his framework of detailed models suitable for process execution. This example fits into the collection of approaches which on the one hand mention the objective of "… bridging the gap between business process engineering and software engineering.", but on the other hand do not explain how this can really be achieved. In addition, the examples presented are "free flying" single processes, with no nested processes or other demanding system structures (see Kloppmann et. al. (2005) for more details concerning the yet unresolved question of how to integrate sub-processes in the WS-BPEL 2.0 specification). And, as already mentioned, they do not practically demonstrate the integration of different levels of abstraction into a stringent methodology.

The repository of assets (service providers, realizing components) as well as other relevant information (data or binding types) is available in the low-level system views and is 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. The choice for UML 2.0 Activity Diagrams was made especially because a unifying technique suitable for all abstraction levels was defined as a core requirement for the methodology. Other *Model Representation Languages* (BPMN, XPDL) consider only single process models.

BPMN 1.0 submitted by BPMI in May 2004 was initially thought as the "natural visual form" for BPML, although the development of the two ran as independent efforts in separate workgroups. In BPMI (2004), the original charter for the BPMN initiative, it is stated that

> *The purpose of BPMN is to be an intuitive notation for the development of BPML processes at the business level. Where BPML is used to carry process semantics among computers, BPMN carries it among business users.*
>
> *BPMI (2004)*

This narrow view has been given up due to the unsuccessful BPML, and now BPMN is promoted as a general process notation which can be mapped to BPEL 1.1 (the specification includes mapping suggestions of certain shapes and patterns to specific BPEL code) as well as XPDL 2.0 specification.

> *XPDL 2.0 is intended to be used as a file format for BPMN. The original purpose of XPDL is maintained and enhanced by this second version of the specification. The XPDL and the BPMN specifications address the same modeling problem from different perspectives. XPDL provides an XML file format that can be used to interchange process models between tools. BPMN provides a graphical notation to facilitate human communication between business users and technical users, of complex business processes. There are a number of elements that are present in BPMN version 1.0 but were not present in XPDL version 1.0. Those had been incorporated into this version of XPDL.*
>
> *WfMC (2005)*

Hence it can be seen as a competing standard to UML 2.0 or other graphical notations capable of process definition. Although the initial intention was to present business analysts with a more user-friendly tool compared to the technical and complex UML, the comparison of White (2004) between BPMN Business Process Diagram and UML 2.0 Activity Diagram shows that there is not much difference between the two, regarding the technical ability to represent the 21 workflow pattern defined by van der Aalst et al. (2000) as well as their readability. It is not unlikely that the OMG will take the BPMN standard, which could eventually result in a consolidation of BPMN Business Process Diagrams and UML Activity Diagrams.

Other mapping requirements, which would gain importance if different modeling notations and concepts should be integrated, are in our case not relevant due to the limitation of modeling environments (Rational Software Modeler 6.0.1 with profile plug-ins and BizTalk Orchestration Designer 2004).

Out of the lists of tasks in Figure 37 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 MS 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).

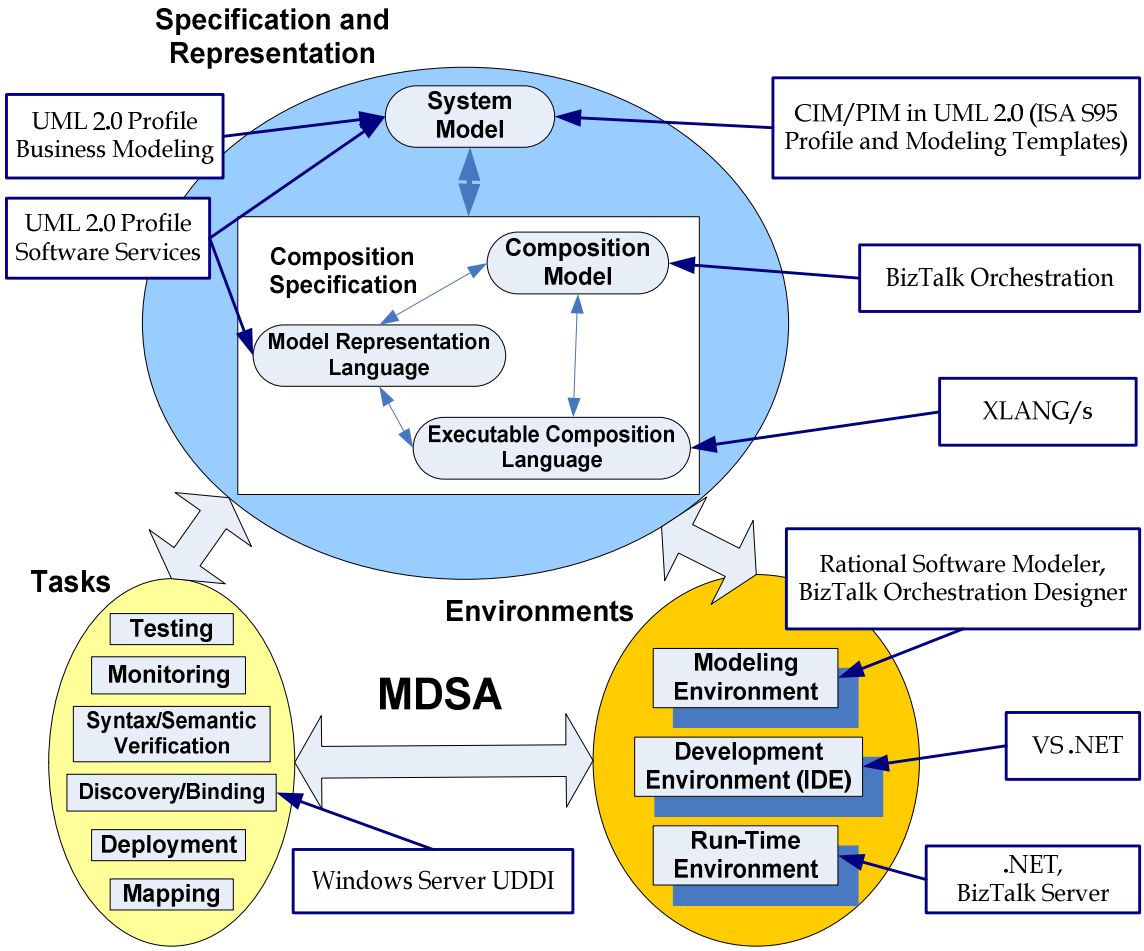


Figure 37. MDSA demo scenario

# 7. MDSA Implementation and Evaluation by Means of a Demo Scenario

In the chapters below the demo-scenario will be described, following the MDSA methodology. It has already been discussed why UML was chosen to implement the methodology. Because a detailed discussion of UML 2.0 is out of the scope for this work, it is assumed that the reader is familiar with UML basics as well as the major changes in Version 2.0. Throughout the text some short language related explanations are given, but in general the figures included, together with the description, should be sufficient in order to understand the models, even with limited UML knowledge.

A good overview of Version 2.0 improvements in the context of model driven development support is given by Selic (2005).

Italic text marks UML stereotypes derived from the profiles, ANSI/ISA 95 and modeling methodology specific phrases.

## 7.1. The ANSI/ISA S95 Meta-Model

Within the modeling framework the complete ANSI/ISA 95 standard can be visualized and used as the guideline for high-level analysis modeling, hence the methodology implementation follows ANSI/ISA 95 Part 1 concerning the general assumptions about hierarchies and functions as shown in Figure 39 and Figure 40. The object model defined in Part 1, and enriched with attributes in Part 2, is not extensively used in the demo-scenario and is therefore not completely covered in the first version of the *Shop Floor Tool-Box* and the corresponding profile. Hence the functions and related information flows of interest are applied in the ANSI/ISA 95 meta-model, but not the complete data type definitions for Level 3/Level 4 information exchange (object models). Nevertheless, the terminology used to name the objects is *always* consistent with the standard. According to that, the *Production Schedule* object released from Level 4 does not comply with the standard regarding the data type definition. Other *Categories of Information Exchange* (Figure 38) do not take place at this interface level in the demo scenario.

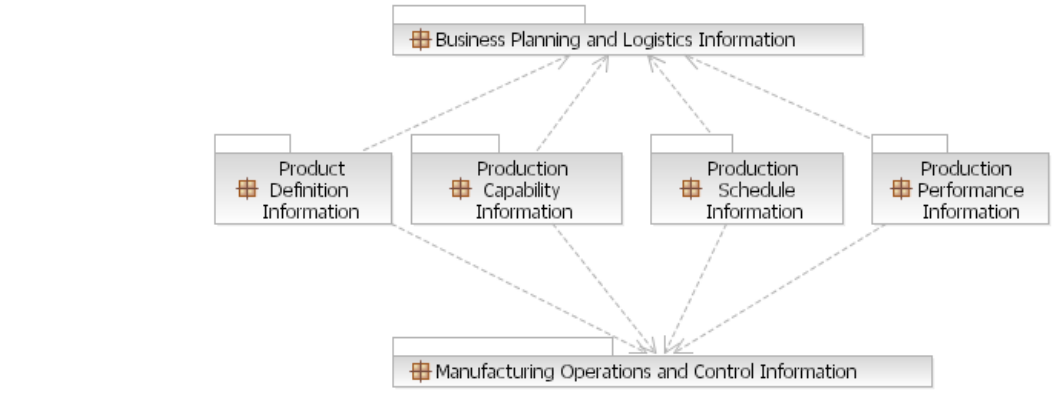


Figure 38. ANSI/ISA 95 Categories of Information Exchange

The generic *Functional Hierarchy Model* (included in Figure 40) and the *Equipment Hierarchy Model* is the base 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*, depicted in Figure 40.

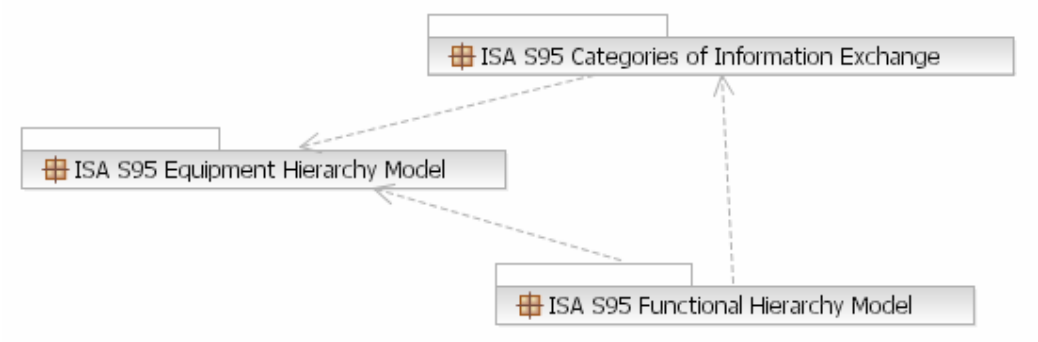


Figure 39. ANSI/ISA 95 Analysis Model Overview

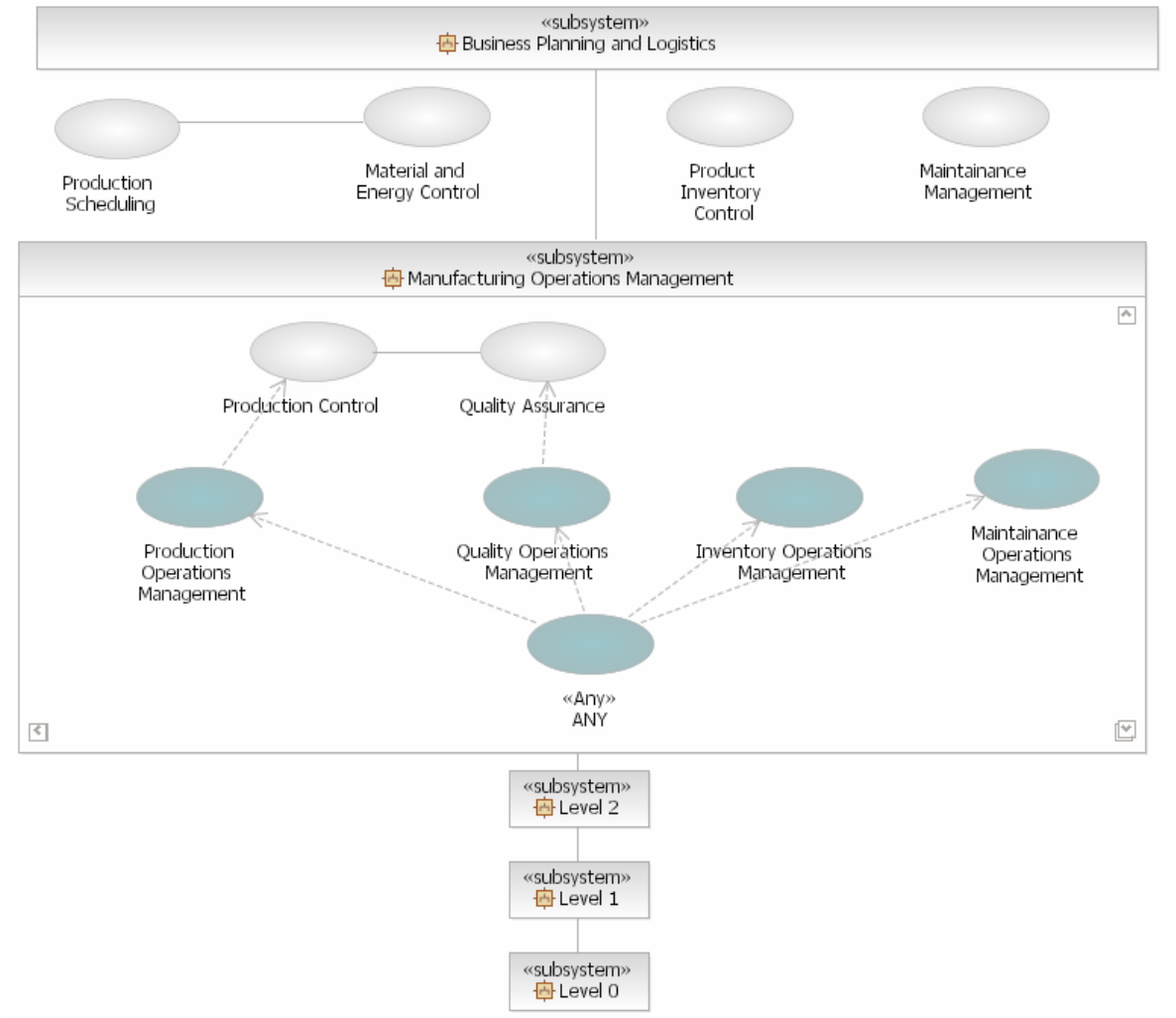


Figure 40. Manufacturing Operations Management model

This use case view shows a high level grouping of activities related to four main operations management areas by means of "include" and "extend" relationships. According to the standard, some activities can be uniquely assigned to the subsystems, derived from a multi-level hierarchy of activities (due to a better visibility the activities of subsystem *Business Planning and Logistics* and their associations with subsystem *Manufacturing Operations Management* are hidden). Others, like *Production Scheduling,* could be assigned to both subsystems depending on the particular scenario. Moreover, the standard supposes supporting activities, like:

- Management of security within manufacturing operations.
- Management of information within manufacturing operations.
- Management of configurations within manufacturing operations.
- Management of documents within manufacturing operations.
- Management of regulatory compliance within manufacturing operations.

- Management of incidents and deviations.

In the model this extension possibility is foreseen similar to XML schema definitions by means of a use case stereotyped with *Any*.

Figure 41 depicts the activity model of *Production Operations Management,* derived from a generic activity model for all four types of operations management (Figure 40). An UML activity diagram was chosen to visualize this model. An UML Activity consists of a number of Actions, which can be of general, of call behavior or call operation type. The Call Behavior Actions provide a behavior property which points to a diagram describing the expected behavior of this action. The ANSI/ISA 95 meta-model does not contain any authoritative behavior models, but in the SFTB and therefore in the demo scenario the Call Behavior Actions point directly to the basic flow diagrams or activity diagrams of the ANSI/ISA 95 activities. Thus each action becomes an activity and therefore nested levels of functionality can be created.

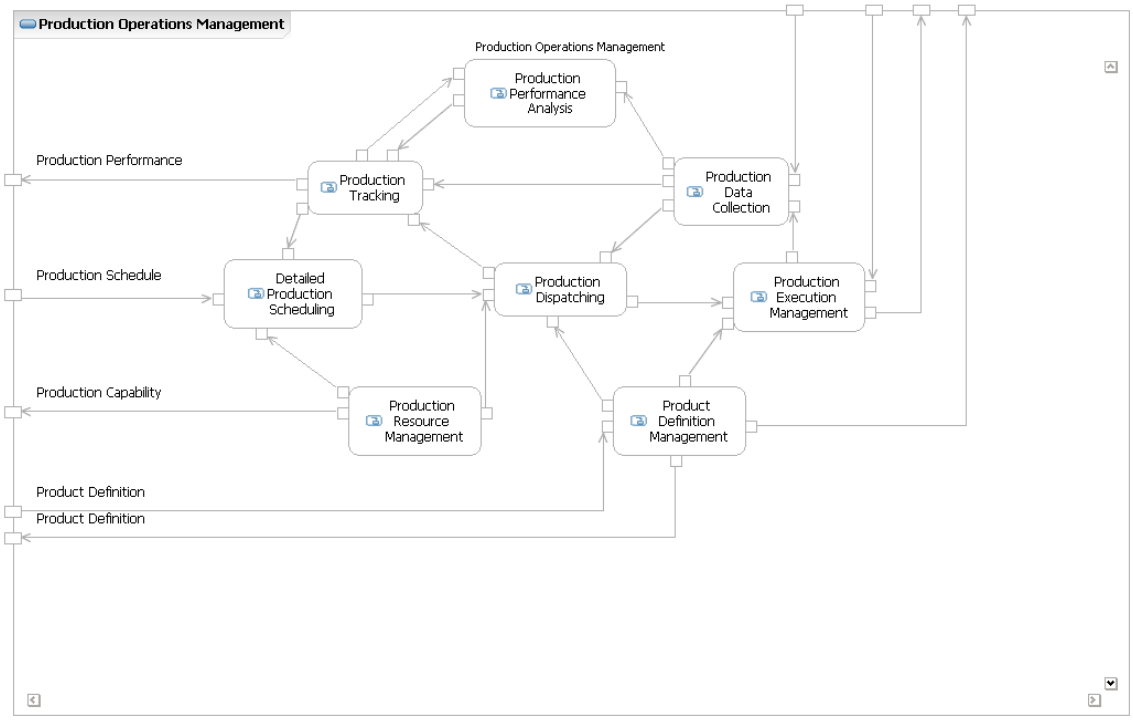


Figure 41. Activity model of Production Operations Management

Keep in mind that at the highest level the UML Call Behavior Actions are ANSI/ISA 95 activities and that the complete *Production Operations Management* model is covered by means of a single UML Activity. All the squares and lines represent the basic object flow as defined in the standard. For external interaction the Activity Node Parameter notation is used. On the left side the Level 4 interfaces are visible. Hence the parameter names used here are members of the *Categories of Information Exchange* in Figure 38. Interfaces to the Level 2 or Level 1 functions are indicated at the top of the activity frame. No strict interface definition for this level is incorporated in the standards, just some typical interactions are mentioned for corresponding ANSI/ISA

95 activities. For instance an object flow from Level 2 to the *Production Data Collection* activity can be found in some form in every shop floor domain to gather information about resource usage and deployment of staff. Again for the meta-model this information has not been included, but for the SFTB this information will be covered. It must be said that object flows between the UML Call Behavior Actions via Input und Output Pins are not consistent with the detailed ANSI/ISA 95 activity models discussed in the following, because the standard does not restrict the creation of object flows where appropriate.

The reason lies in a recommendation like formulation, thus the standard continuous with an informal discussion of the functionality of the ANSI/ISA 95 activities depicted in Figure 41 and names data objects exchanged at some possible interfaces. This information is fully covered in the meta-model and is completely available in the *Shop Floor Tool-Box*. But the real aim of this thesis is to close the gap between Business and (Software) System Engineering, to define and detail the tasks (services provided) of the single activities above, map the roles to service providers, which again are physically bound to personnel or applications/systems. To give an example: The *Detailed Production Scheduling* activity has an associated role *Scheduler*, which has to fulfill parts of the service flow *scheduleNewOrders*. This role is assigned to the scheduling software Preactor as well as to personnel.

The MDSA approach introduced earlier helps to handle the process from high-level system models to executable service flows by means of modeling, therefore the corresponding views from the modeling environment are included in the scenario description where appropriate. The demo scenario helps to visualize the introduced methodology, starting with the initial, particular shop floor model creation by means of the SFTB, and ending with a run-time execution environment ready to step into the MDSA control loop.

## 7.2. Shop Floor Tool-Box Implementation

ANSI/ISA 95 is on the one hand the high-level framework for the SFTB, a collection of models and artefacts ready to be used for modeling projects. Thus the first task was to build an ANSI/ISA 95 UML meta-model in the modeling environment. The models derived directly from the standards and were introduced above. On the other hand this meta-model constitutes the UML Profile for ANSI/ISA 95, which is used throughout the modeling efforts. Mainly, the purpose of this profile is to keep the relationships to the 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 standard representation. It contains more information at different levels of abstraction and shall grow with every real-world modeling project, which means entities like service providers, data objects etc. are continuously played back in the SFTB.

For the framework organization, the Rational Analysis Model template is used. *Functional area* packages are introduced at the top level of the model. Each *functional area* package contains analysis-level use-case realizations for a particular functional area plus the analysis classes that play roles in those realizations. Each use-case realization is represented through an UML Collaboration. Use cases and actors are packed in a separate use-case package. This separation of use case model and analysis model can also be found in the IBM (former Rational) UML Profile for Business Modeling (Johnston 2004, Ng 2002, Rational 2001). The semantic necessary in the business context is introduced by means of this profile. The before mentioned *Analysis Model* template provides the basic framework for fast business modeling because it is following the structure of the profile. Due to the integration of the *Business Modeling* profile into the template the analysis classes *Entity* and *Control* derived from the *RUPAnalysis* profile referenced in the template model get the additional stereotypes *Business Entity* and *Business Worker* respectively. For the graphical representation in the diagrams the latter notations are used. The *Boundary* class has no equivalent in the *Business Modeling* profile and thus remains unchanged.

### 7.2.1. SFTB Generic Entities beyond ANSI/ISA 95

The SFTB is a knowledge repository, growing with every particular domain modeling project undertaken. The ANSI/ISA 95 meta-model represents the generic level framework which will be extended by means of generic as well as particular entities. Some of the already designed information groups will be introduced in the following. Next to the corresponding UML types and diagrams, icons are used as often as possible to express the semantic of each entity as clear as possible. Each group is held in an UML Package at the proper level of abstraction.

- Connection Types
- A broad range of connection types are needed for the shop floor domain, starting with paper or floppy disk exchange and ending with sophisticated communication standards like OPC Data Access. The latter is available as a class diagram showing the interfaces entities like OPC Server or OPC Client request and provide.
- Data Types
- Some data types are already incorporated as ANSI/ISA 95 suggestions for data exchange between Level 3 activities. Much more were added in the SFTB as UML classes, partly arranged in a class diagram to show the dependencies and relationships between them. Attributes are included, but are likely to change in a particular scenario. Nevertheless, it proved useful to have a suggestion as a starting point. First, some data fields are almost always necessary and at most the name changes (e.g. an order ID field in a work order data type). Secondly, knowing different types of e.g. work orders in particular scenarios make best practice comparisons possible. Next to general shop floor data types for basic operation management, additional entities for selected information domains

were included in the form of UML Packages. For instance, there is a tool package consisting of class diagram and tool information related classes. Vendor application specific data formats as well as standards organization body proposals (e.g. OAGIS 9.0 Business Object Documents, MIMOSA, or B2MML) could be added as well.

- Business Goals
- Goal hierarchies and their relationship with core processes are valuable and reusable information incorporated in the SFTB
- Process Definitions
- In the SFTB complete business processes can be stored, either platform independently or platform specifically.
- Service Providers
- Service Providers are the last piece of information missing to build and maintain platform independent high level information architecture packages in the SFTB, for instance the aforementioned tool package or a DNC package. If a particular model is demanded, it can be customized according to the given situation.

## 7.3. Production Operations Management Demo Scenario Activities

In Figure 41 the generic activity model defined in the ANSI/ISA 95 standard can be found, whereas in the model presented in Figure 42 only those activities actually implemented are included (particular model). As one can see, this enables a direct high-level as-is/to-be comparison regarding ANSI/ISA 95 compliance.

The activity diagram gives a complete high-level overview of the functionality and the basic object flow available in the system. The further decomposition down to executable code is fully supported by means of the *MDSA for the Shop Floor* methodology which is described in the following. Let us first take a closer look what is happening in the demo-scenario. First of all it must be stated that at this level the control flow is completely hidden, thus it can not be said in which order the interactions will take place. For that matter flow diagrams have to be added in a subsequent step. The information flow is restricted to the major contributions of each activity and lack the detailing of interaction patterns or the like. Third, this level is a kind of black box architecture, thus it remains unclear how much functionality is available in each UML Action (ANSI/ISA 95 activity). Moreover, all functionality outside the scope of the manufacturing information standard is missing as well, for instance infrastructure functionality like that provided by a repository.

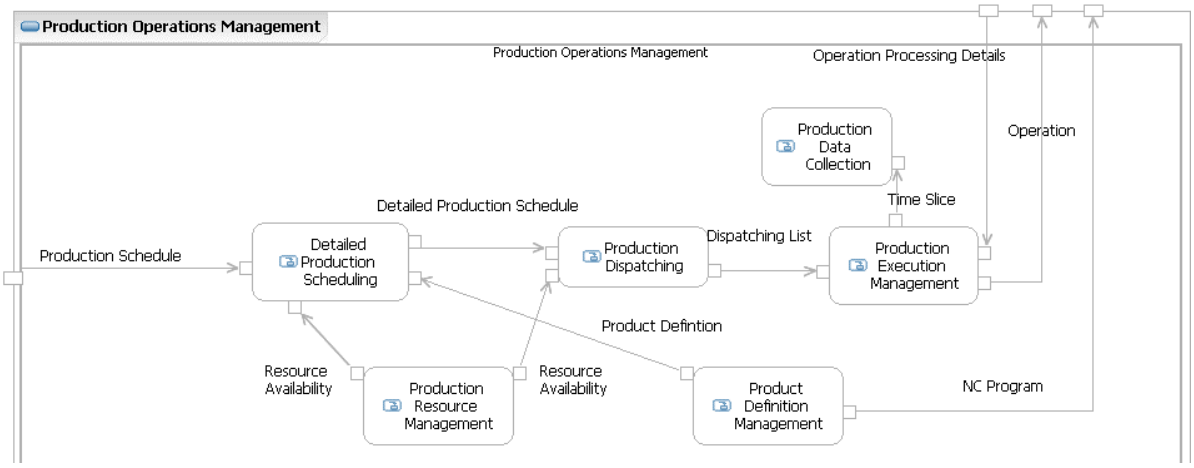


Figure 42. Implemented Production Operations Management activities/object flow

An executive summary using this model could look like this:

To manage the production operations between our ERP system and the machinery, as well as the workers at the assembly line in plant 1, a *Production Schedule* has to be released from the ERP system rough planning module. The schedule will contain only the order data, but not the product structure. The *Detailed Production Scheduling* activity needs the *Product Definition* (mainly Bill of Material information) of the requested products, thus this data has to be provided by the *Product Definition Management* activity. To schedule the operations according to the order and BoM structure, target resources have to be defined. This resource related medium-term information is retrieved from the *Production Resource Management* activity. After the *Detailed Production Scheduling* activity has performed the scheduling task, a *Detailed Production Schedule* will be the result. This object represents the input into the *Production Dispatching* activity which will release resource specific dispatching lists according to the short-term resource availability. *Production Execution Management* will provide the operations contained in the *Dispatching List* to the resource and waits for operation processing details in return. Certain details will be passed to the *Production Data Collection* activity as *Time Slice* objects. As NC machines are one type of resources, *NC Programs* have to be provided by the *Product Definition Management* activity as well.

So this is the *as-is* system. Of course we do not know the system in detail, but if one compares the generic activity model in Figure 41 and the particular in Figure 42, it will be obvious where improvement is possible. With the help of the generic model enriched through control and information flows deposit in the SFTB a *to-be* scenario evolves. It could look like the one in Figure 43.

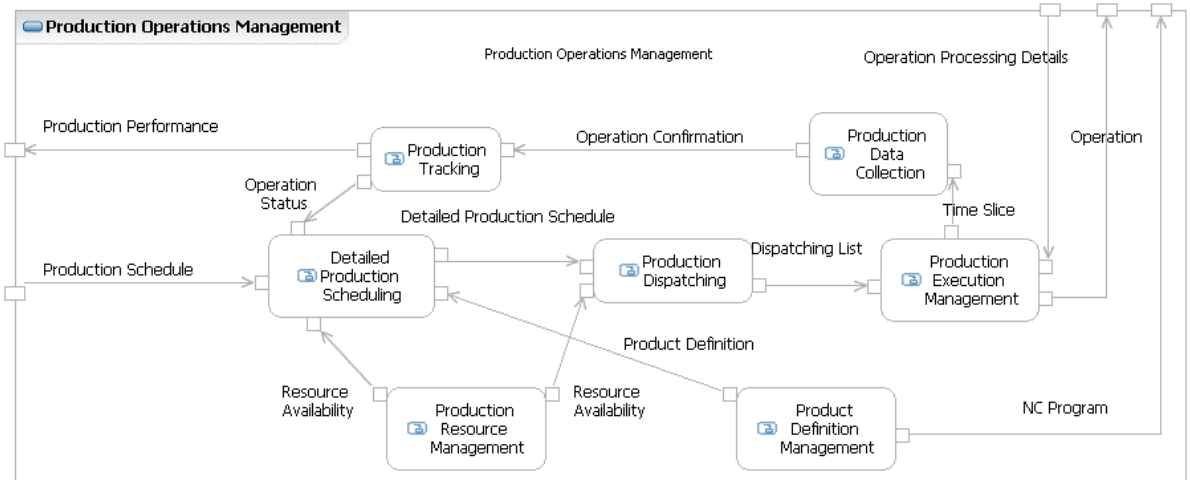


Figure 43. TO-BE production operation management scenario

The improved information system tackles the weak point of the given system regarding the information flow upwards. In the present situation the *Production Data Collection* activity deals with fine-grain data objects which are not suitable for upward processing. But the *Detailed Production Scheduling* activity needs a closed control loop if reactive scheduling is a new goal (goal setting is discussed in the following). Therefore the actual operation status is necessary, provided by the *Production Tracking* activity. There the operation confirmations, which are consolidated time slices, are transformed twofold: First as *Operation States* suitable for the scheduling activity and second as *Production Performance* data suitable for the Level 4 functions. Different data types and different cycle times will probably be the main distinctions between the two.

Figure 44 continues the detailing of the above functional model for *Production Operation Management* for our demo scenario SOA. It could be argued that the use of the ANSI/ISA 95 activities to perform the decomposition is exaggerated for the given system and that the basic flow would very likely fit into a single flow diagram as well. This may be correct for primitive information architectures, but for complex systems next to the vertical organization through abstraction levels, some kind of horizontal organization is needed as well. The agreement on this horizontal organization framework by means of ANSI/ISA 95 activities, is the major achievement of this standard. Of course in the end the reassembled activity models must show a consistent picture of the whole architecture.

The *Business Analysis Model* establishes the organizational framework to decompose the ANSI/ISA 95 activities. Six *Business Analysis Models* (one for each activity) import the corresponding parts of the *Production Operations Management* use case view, which is consisting of a single diagram. The use case view contains the use case models like the one depicted in Figure 45, so it represents a single point of information for all *Business Use Cases* as well as *Business Actors* and *Business Goals* for the activities.

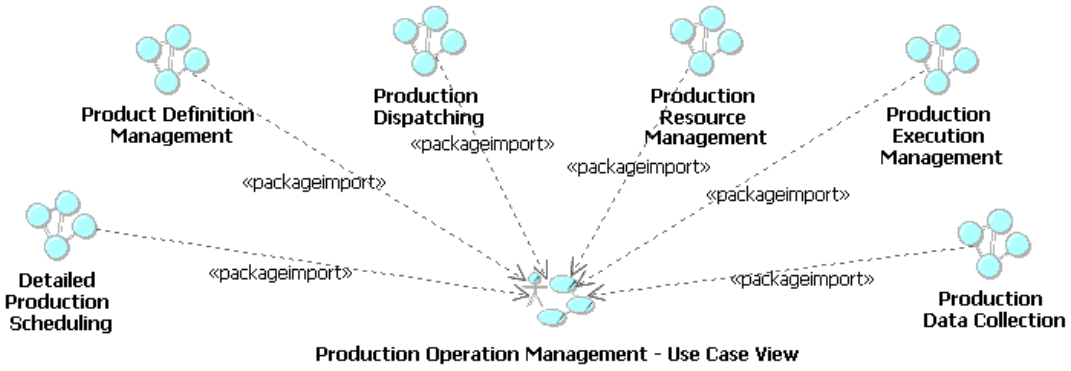


Figure 44. Business Use Case and Business Analysis Model main diagram

The decision to separate the use case models from the rest of the content of the *Business Analysis Models* was made due to possible dependencies respectively intersections between use cases, actors and goals. It could be argued that this network should be represented in a single diagram. But the use case realizations will reference the use cases to make the relationships between use cases and realization visible in the *Business Analysis Model* again (Figure 46).

For the remainder the Detailed Production Scheduling model shall be used to demonstrate the easy and highly integrated modeling methodology down to executable process specifications. All other activity models are described textually and figures are only included were they conveniently enhance the understanding of the subject matter.

### 7.3.1. From SFTB Entities to Particular Shop Floor Models

> *Detailed production scheduling shall be defined as the collection of activities that take the production schedule and determine the optimal use of local resources to meet the production schedule requirements. This may include ordering the requests for minimal equipment setup or cleaning, merging requests for optimal use of equipment, and splitting requests when required because of batch sizes or limited production rates. Detailed production scheduling takes into account local situations and resource availability.*
>
> *ANSI/ISA (2005)*

The concept of modularization has been important in the SFTB from the very beginning. That is why the use cases are separated from each other (Figure 45). This gives the modeler the flexibility to chose the entities he needs first 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.

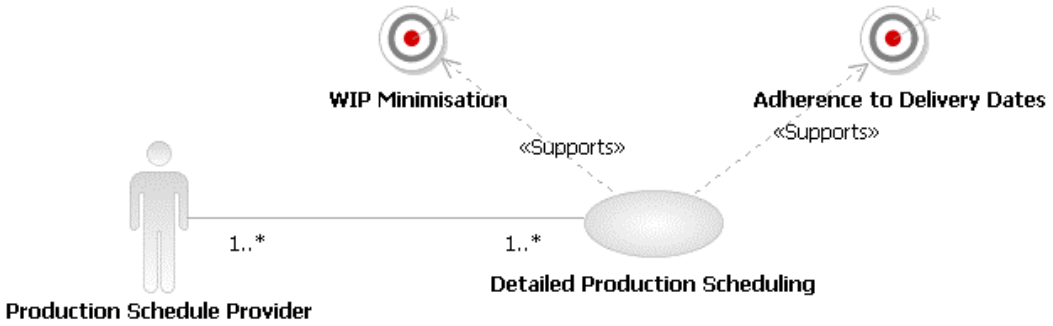


Figure 45. Detailed Production Scheduling Use Case Model

The figure above represents the highest activity level of 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 the MRP II scheduled *Production Schedule* (Kedir Biadgligne et al. 2004). There can be more than one schedule provider (e.g. the ERP system releases a complete Production Schedule, whereas an ATP system just finished product demands) and more than one *Detailed Production Scheduling* use case (e.g. in a multi-site scheduling scenario, multiple Scheduling systems of distributed factories have to interact to fulfill a single *Production Schedule* released for the production network). This use case supports two *Business Goals*, *WIP Minimisation* 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 (Key Performance Indicator), 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 entities derived from the SFTB.

In Figure 46 the main diagram of each of the *Business Analysis Models* of Figure 44 can be seen, showing which *Business Use Case Realizations* (UML Collaborations) actually realize the use cases. From here the link to the overview diagram allows us start explaining the single business level realizations of the use case. In our case a single collaboration is sufficient.

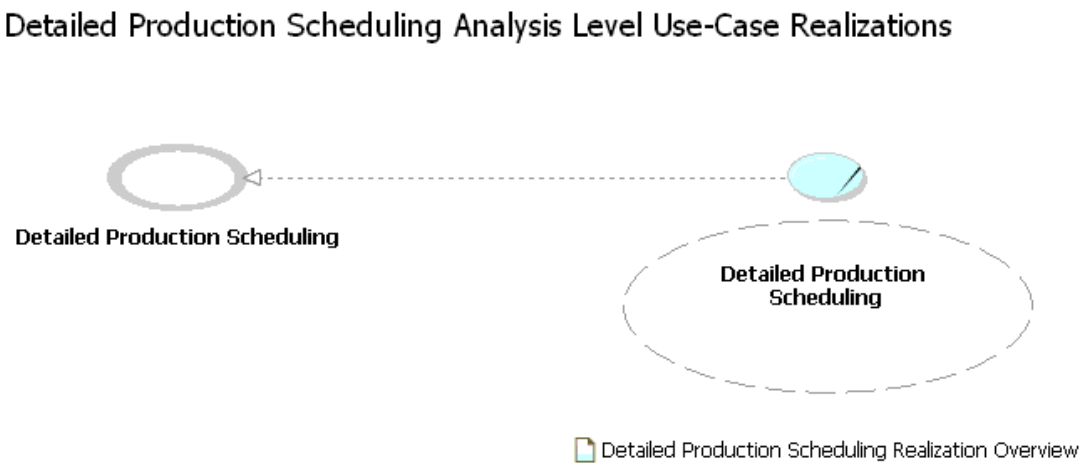


Figure 46. Detailed Production Scheduling analysis use case realization

Following the link leads to the *Detailed Production Scheduling Realization Overview*, a diagram consisting of four diagrams according to the *Rational Analysis Model* template:

- Detailed Production Scheduling Analysis Classes (UML class diagram)
- Detailed Production Scheduling Participants (UML class diagram)
- Detailed Production Scheduling Basic Flow (UML sequence diagram)
- Detailed Production Scheduling Alternative Flow (UML sequence diagram)

But alternative flow diagrams are optional and of course their number is unlimited. The template suggests using sequence diagrams, but the use of activity diagrams or even non-UML diagrams is possible. This overview diagram shall be the central hub for all information regarding the use case and therefore ANSI/ISA 95 activity realization at the business level.

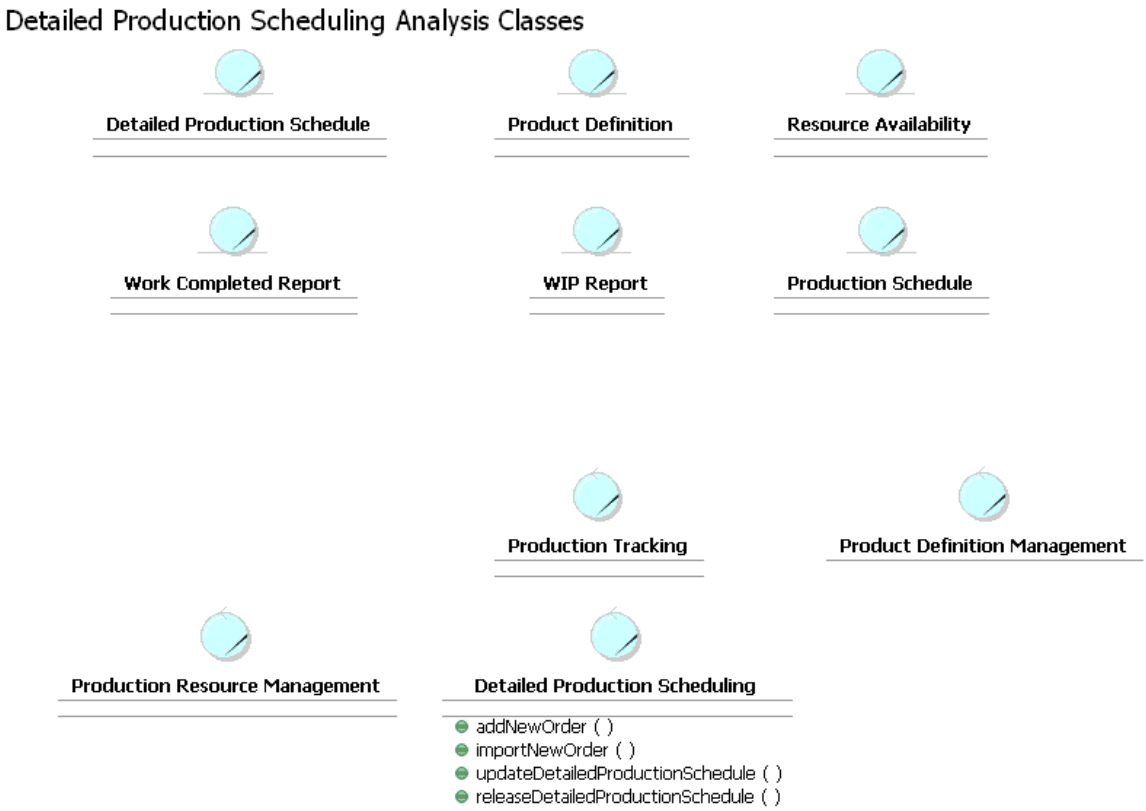


Figure 47. Detailed Production Scheduling Analysis Classes

The first diagram is a class diagram depicting all *Business Workers* and *Business Entities* for this particular realization. In the modeling environment they are collected in a UML Package stereotyped *Business System*. Within the complete set of six *Business Analysis Models*, *Business Workers* are unique. This means, every *Business Worker* belongs to a single *Business Use Case Realization*. All additional occurrences in other *Business Analysis Models* of ANSI / ISA 95 activities are only references. The purpose of this decision is to make the consequences of one missing *Business Worker* immediately visible in the static and dynamic views in every single Analysis Model. Explained by means of our example, this means that the *Detailed Production Scheduling Business Worker* (N.B.: the definition of this stereotype says that it marks a role) is definitely a part of the *Detailed Production Scheduling Analysis Model*, but not the other three *Business Workers*. To make this even clearer, the operations defined are only visible for the classes in their unique view. So if the *Production Tracking* activity is not implemented and thus is missing as a *Business Analysis Model*, the *Business Worker* would be missing in Figure 48 and in every other high-level realization model.

The lower part of Figure 47 groups the *Business Entities* which are used in the scenario. Figure 48 gives a high-level, static view of the role itself, its interactions with other roles and the data objects passed between them.

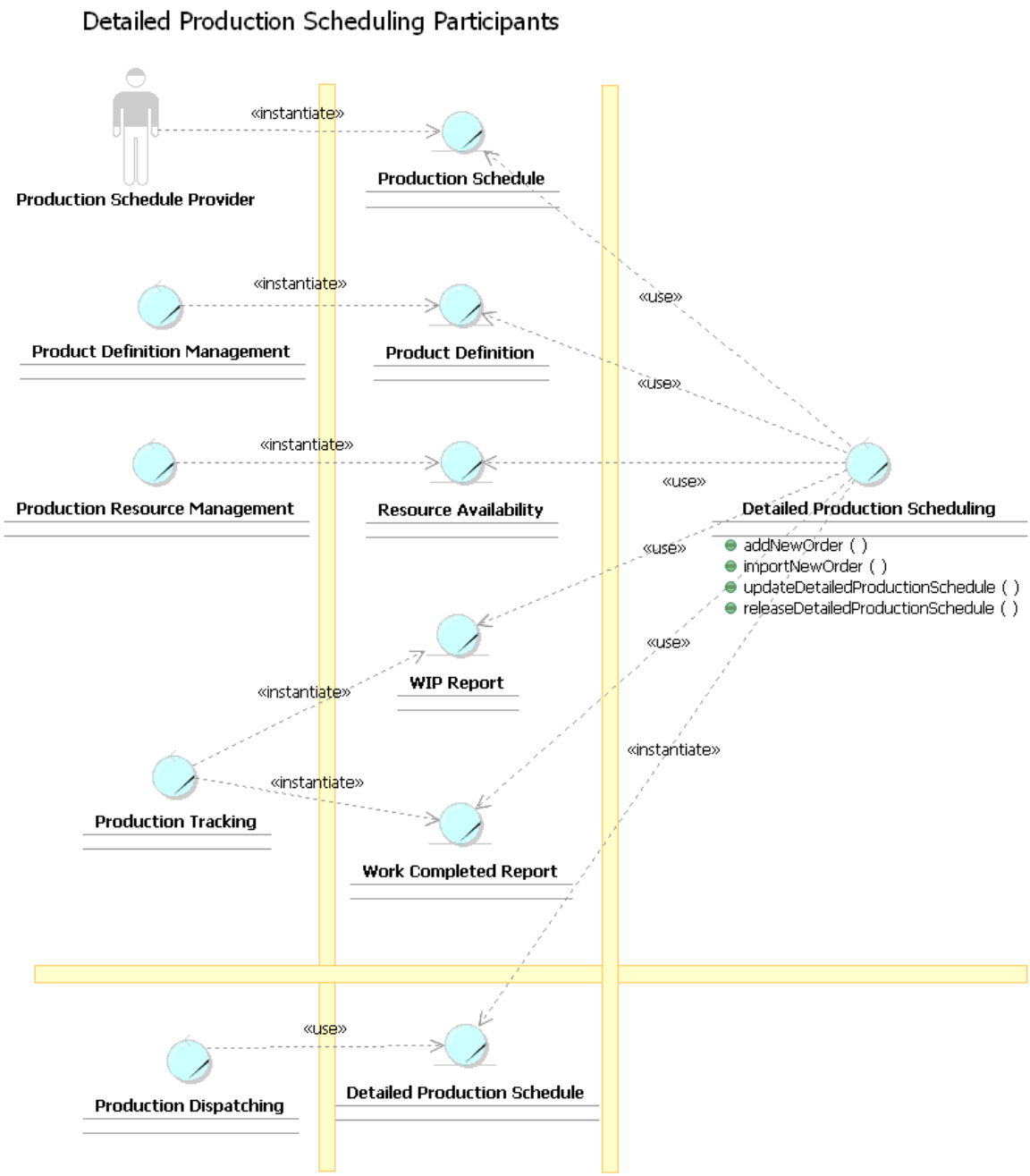


Figure 48. Detailed Production Scheduling participants

It is the second compulsory diagram in every *Business Analysis Model*. At least one dynamic behavior view completes the basic overview. Following the template a sequence diagram was chosen (Figure 49). The basic flow diagrams are directly linked to the Call Action Behavior (double click) constructions in Figure 42.

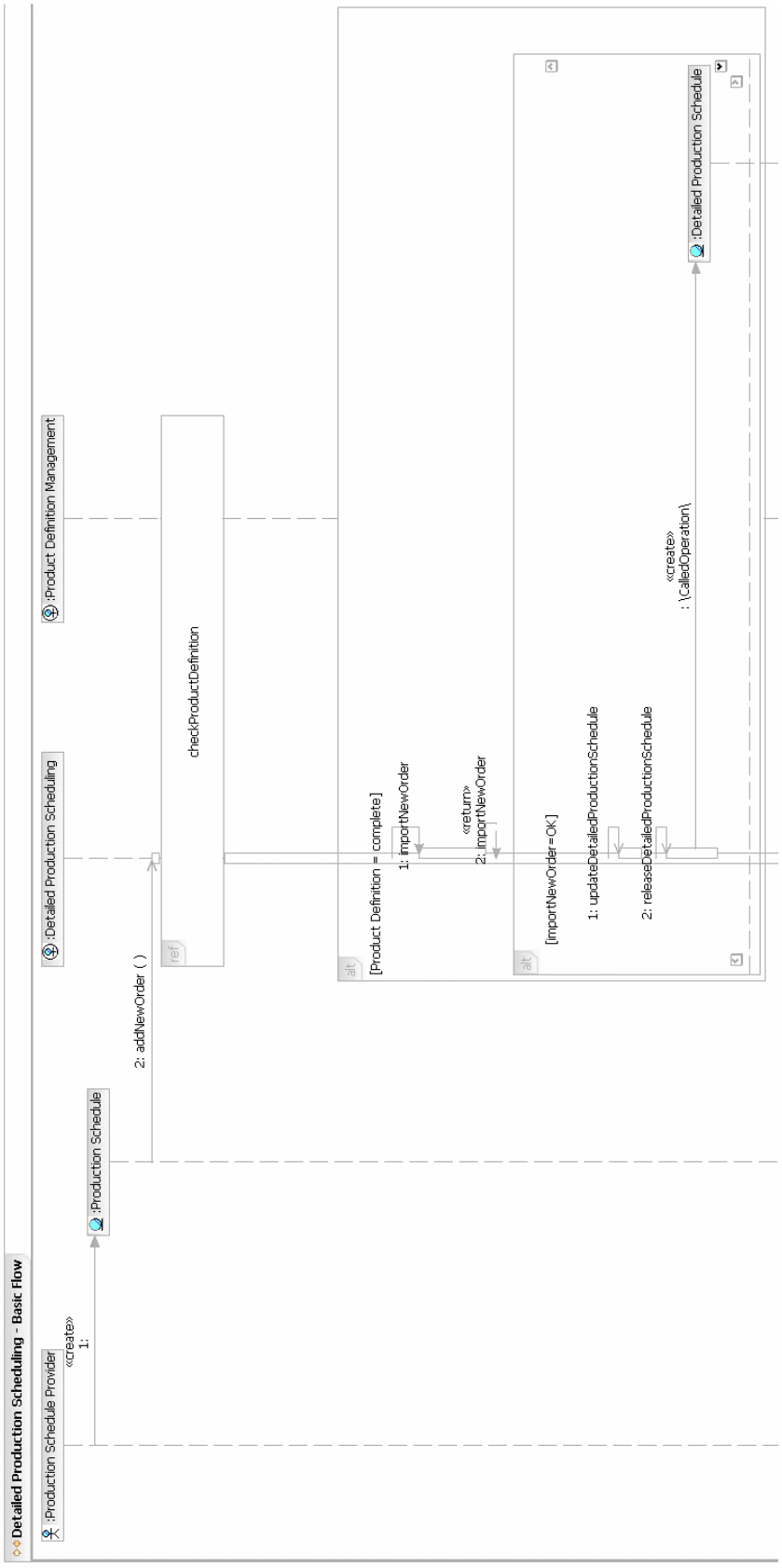


Figure 49. Detailed Production Scheduling basic flow

So far the high-level business modeling capabilities by means of the ANSI/ISA 95 based Shop Floor Tool-Box were presented. The result of the process described up until here is a PIM for our demo scenario, derived from entities of the SFTB. If we continue the top-down approach, the point would be reached go on with the integration of the low-level service layer environment following the IBM Profile for Software Services. But before that we must analyze and model the given functionality in the system in a bottom-up proceeding. After that, the mapping between business layer and service layer can take place.

### 7.3.2. From Particular to Executable Shop Floor Models

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

The *Service Providers* are grouped in a *Service View* perspective (Figure 50). We have already placed an emphasis on the fact that the *Service Provider* construction is implementation independent. That is to say, the *Service Providers* depicted in Figure 50 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*, which participates in the *Service Collaboration scheduleNewOrders*. This role is assigned to the scheduling software *Preactor 9.2* interface extension. Another service provider is *SchedulingProvider*, a role assigned to the Preactor 9.2 software user. The implementation details will be added in the subsequent step of *Component View* creation. Note that this picture does not include all service providers within the demo scenario, but only the ones involved in the *Detailed Production Scheduling* activity.

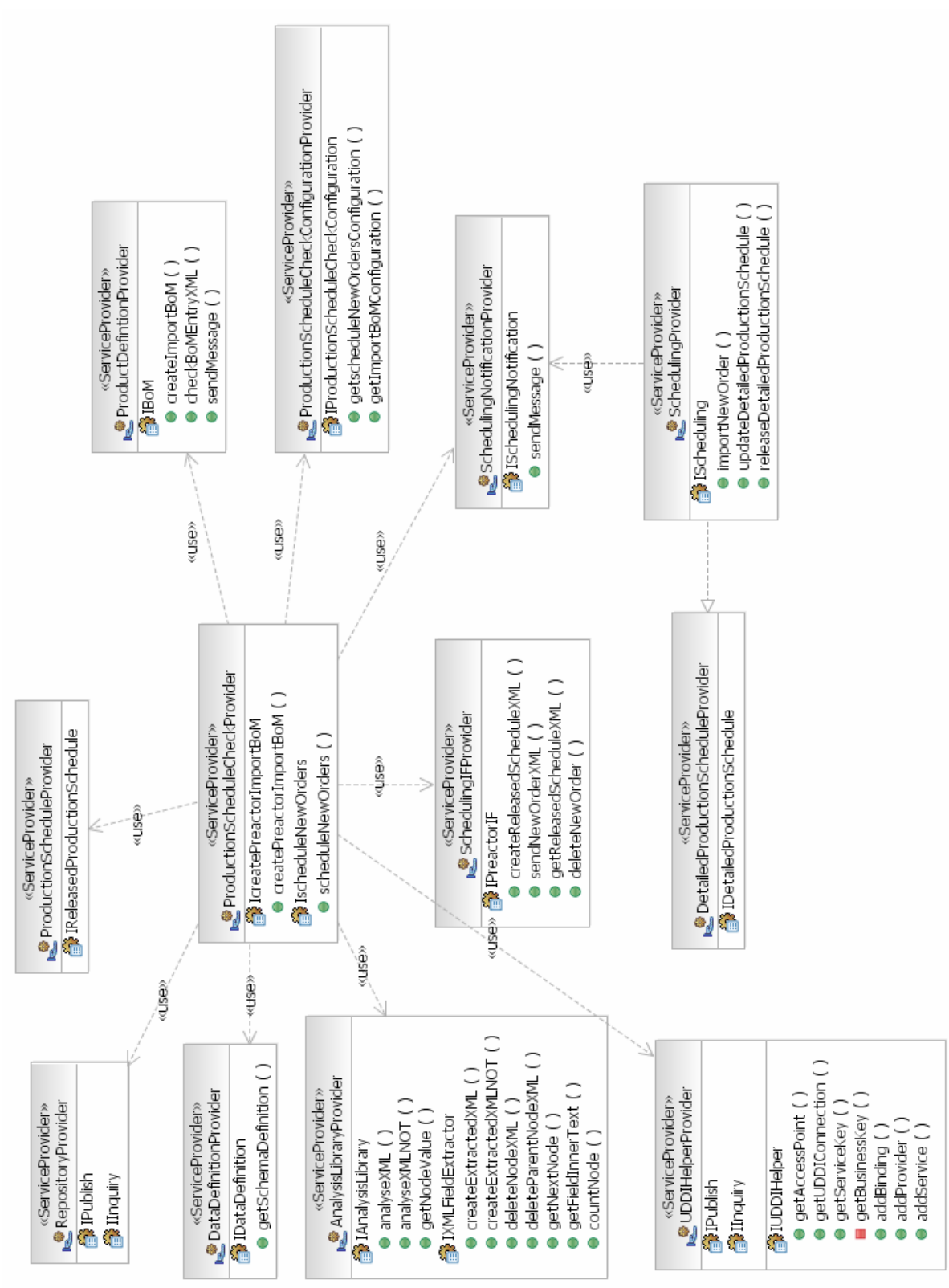


Figure 50. Service View perspective including the service providers for the Detailed Production Scheduling activity

The *Service View*, a component diagram, considers use dependencies as well as realization relationships. A class diagram and a composite structure diagram (Figure 51) refine each single service provider component. For a detailed discussion of all service providers within this demo-scenario we refer to chapter 0. Although some readers may prefer to get an introduction to the system functionality at this point, it is the opinion of the author that the discussion of this extensive issue would distract from the methodology explanation too much. Hence the *ProductionScheduleCheckProvider Service Provider* shall demonstrate the creation of *Service Collaborations* and finally the composition models.

The *ProductionScheduleCheckProvider Service Provider* consists of two Interfaces, stereotyped as *Service Specification*. Which operations they provide can be explored in the class diagram or in the general *Service View*. Each interface can be accessed through two ports (*Services*). Figure 51 depicts this circumstance.

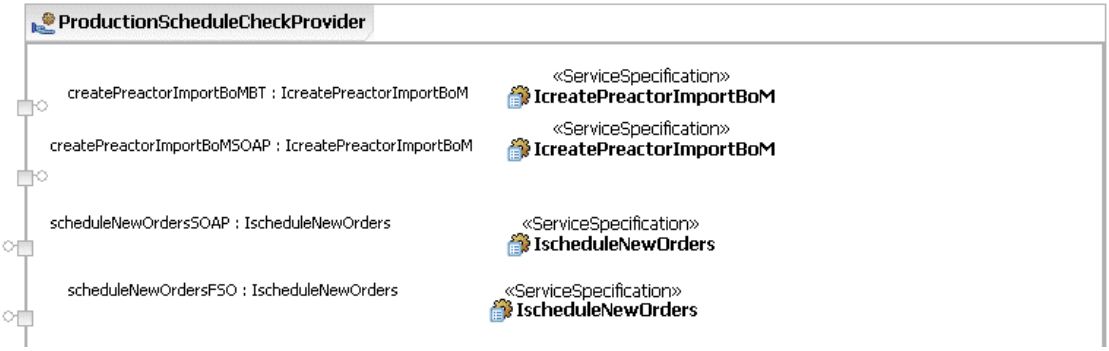


Figure 51. The implementation of the ProductionScheduleCheckProvider

The little squares represent ports, with their names right next to them. Each port is connected to a *Service Specification*. As one can see, at a time two ports have the same *Service Specification*. All in all three port types are used in the whole scenario, namely a SOAP port, a FSO (File System Object) port and a BT port. BT port is a "BizTalk" port, which means that the port only exists as an internal BizTalk interaction channel. This is used to represent internal orchestration calls. The general type is always included in the port name (e.g. scheduleNewOrdersSOAP and scheduleNewOrdersFSO *Services*). The detailed type specification (e.g. SOAPRpc, SOAPDoc) 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 Figure 52, shows the relationships between the *Service Partitions*, a collection of *Service Providers*. It is also shown what roles the *Service Providers* fulfill in the partition (e.g. role *Scheduler* of type *SchedulingProvider*). In our case the partition is compliant with the activities in Figure 42, thus we find a *Detailed Production Scheduling Partition*, which realizes the role of a *DetailedProductionScheduleProvider* for other partitions.

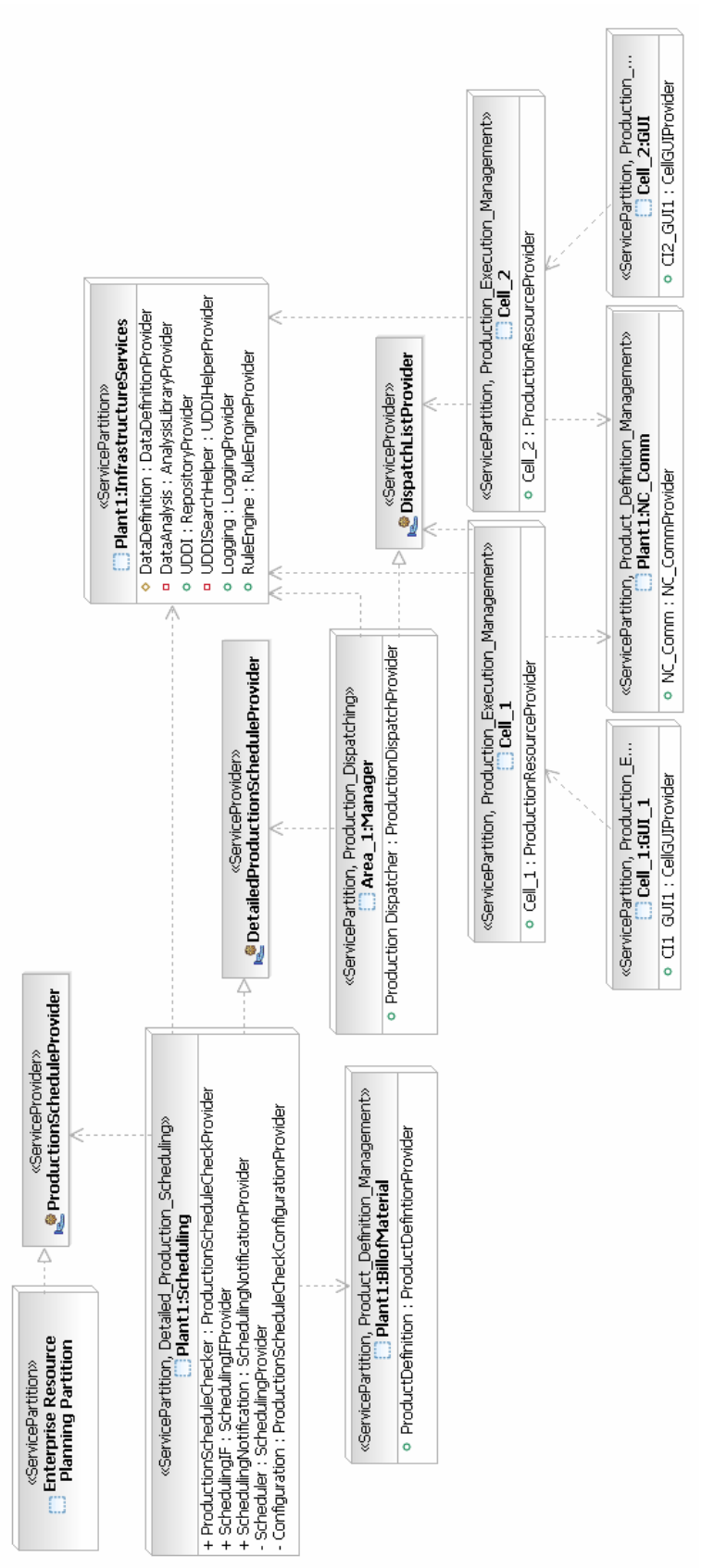


Figure 52. Composition Overview of the Collaboration Overview perspective

The careful reader will soon detect that the *Production Resource Management* and the *Production Data Collection* activities are not represented as partitions. The reason therefore is that resource management is completely incorporated in the scheduling software functionality by means of a resource database and thus it is part of the *Detailed Production Scheduling* partition *SchedulingProvider* implementation. The data collection functionality is restricted to a simple receive message port so far, thus it was decided to exclude it as long as more functionality is available.

Partition and UDDI are closely related. It was decided to map each *Service Partition* to a UDDI service provider and every *Service* operation to a UDDI service due to the small number of operations in addition to the demand for single operation visibility in the UDDI. The UDDI categorization is ANSI/ISA 95 compliant as well, therefore the UDDI provider *Plant1:Scheduling* has the following categorizations assigned to it 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).

So far we only detailed the system functionality by means of *Service Providers* and linked them to the high-level ANSI/ISA 95 activities by means of stereotypes and logical grouping (structure mapping). What is missing so far is the service oriented realization of the operations and flows defined earlier for each activity (Figure 49).

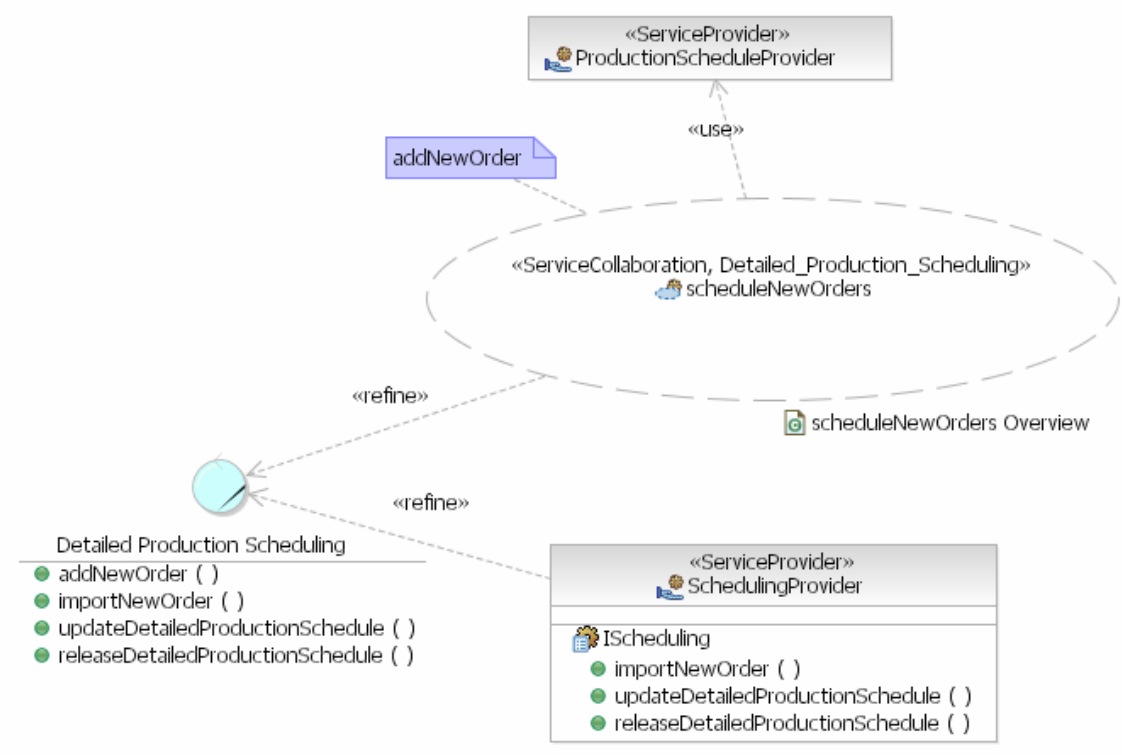


Figure 53. Particular to executable shop floor model mapping

This behavioral mapping between Particular Shop Floor Model activity and Executable Shop Floor Model Service operations is depicted in Figure 53. It is obvious that the former *addNewOrder* operation is now realized by the *scheduleNewOrders Service Collaboration*, which again is an operation provided by the ProductionScheduleCheckProvider *Service Provider* (Figure 50). All other operations invoked in the basic flow are now realized by the *IScheduling Service* of the *SchedulingProvider Service Provider*. Later we will see that these operations are performed by personnel, whereas *scheduleNewOrders* is deployed as a BizTalk orchestration.

In the basic flow the implementation of operations was completely ignored, but at the service level it must be elaborated which is a complex service composed of a number of different services (which can again be basic or complex). Therefore the next selection of diagrams depicts the actual *Service Collaborations*, which again are assigned to ANSI/ISA 95 activities by means of stereotypes. Each collaboration contains a composite structure diagram and at least one diagram for the behavioral view.

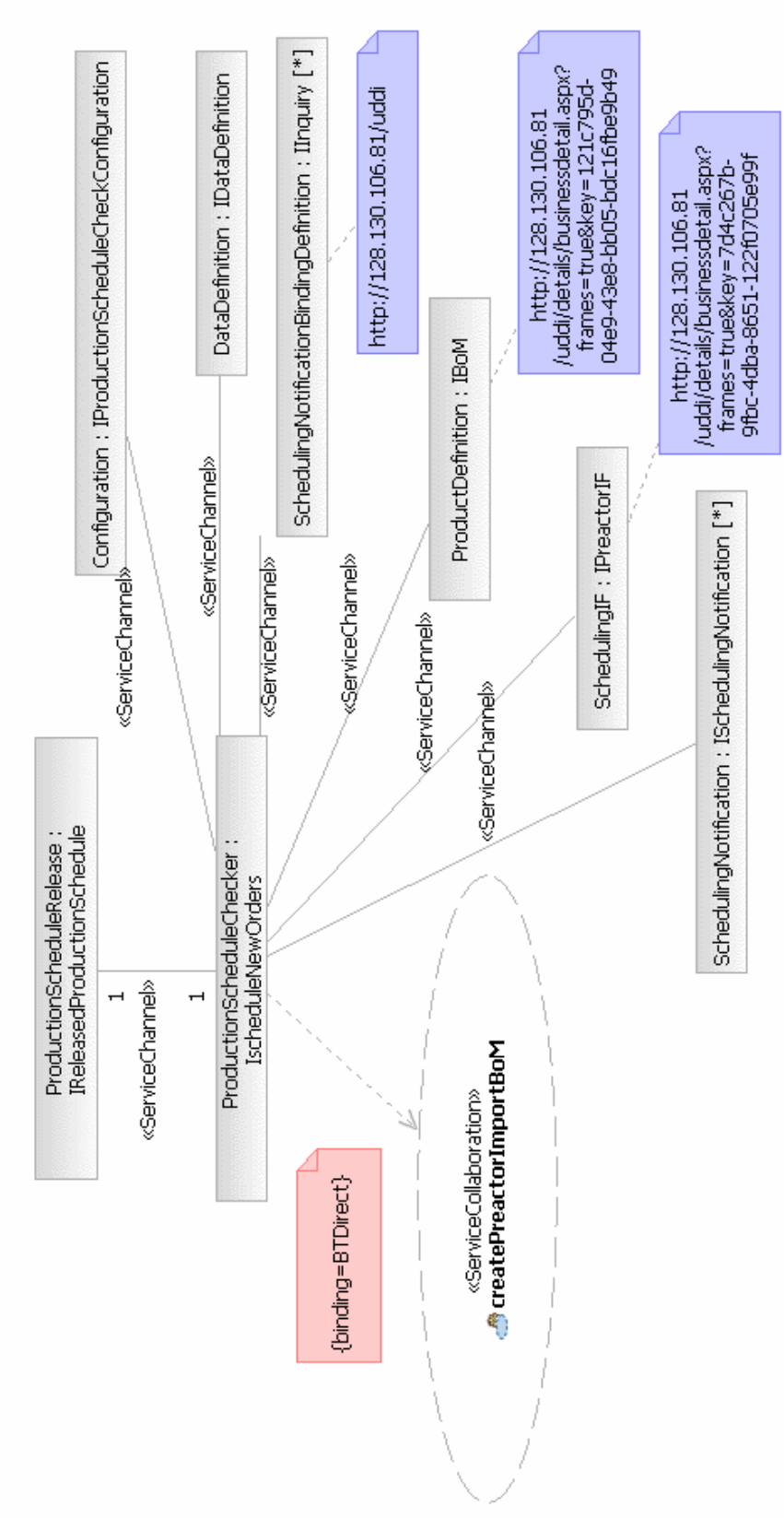


Figure 54. *scheduleNewOrders* collaboration composite structure

Figure 54 depicts the static composite structure of the *scheduleNewOrders* collaboration. Here we see the participating roles and the interfaces (*Service Specifications*) they realize. The *Service Channel* stereotype contains the binding information. For this collaboration we define two binding types, namely SOAPDoc and BTDirect. SOAPDoc specifies WS interactions, and BTDirect interactions within or between BizTalk assemblies. What we also see is that this collaboration depends on another *Service Collaboration*, namely *createPreactorImportBoM*. The nested collaboration gets bound 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 endpoint and 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 subscription in the UDDI takes place (*IInquiry* interface). The *DataDefinition* participant represents the XML scheme definitions used in this collaboration, which are in this case deployed as .NET DLL at the BizTalk Server. 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 as an independent orchestration makes the rule platform become independent, because the "configuration helper orchestration" can be published as a WS, if necessary.

Due to the limitations of space it is only possible to present the upper part of the activity diagram for *scheduleNewOrders*.

Figure 55 depicts the control- and object-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. Nevertheless, it represents the lowest level of abstraction, which means that certain implementation decisions are already included (e.g. the decision to store messages persistently in FSO and not in databases. This is reasonable because the number of flow executions for this collaboration is restricted). Next to the control and object flow comments and business rule constraints can be found. Two types of comments can be identified. First, there are comments which point to the physical location of ports, in this case only to file system objects. Ports can be UML Activity Parameter Nodes, Data Store Nodes and Central Buffer Nodes. While the first represents a public port of the process, the second is a private port which persistently stores messages. The latter is a protected port, which means only entities defined in the particular collaboration can access the port. Remember the collaboration composite structure depicted in Figure 54, where we defined that the *scheduleNewOrders Service Collaboration* is dependent on the *createPreactorImportBoM Service Collaboration.*

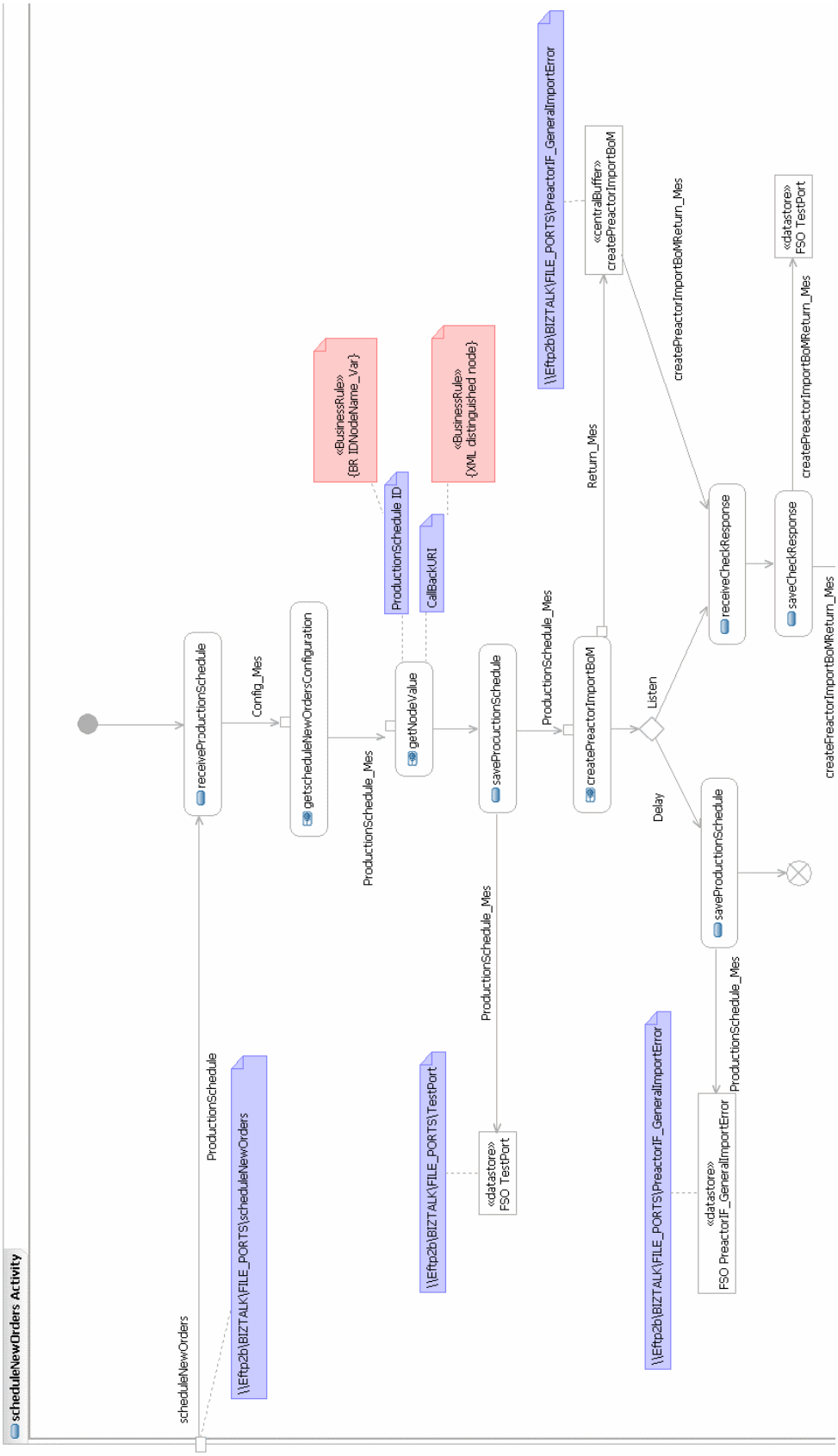


Figure 55. scheduleNewOrders activity diagram (upper part)

In Figure 55 you can see the corresponding UML Call Operation Action with the *Return_Mes* object as output. The control flow can not continue until this object is dropped in the protected port. The same message enters as *createPreactorImportBoMReturn_Mes* message the listening instance *receiveCheckResponse* of *scheduleNewOrders*. If this object is delayed (timeout), then the initial *ProductionSchedule_Mes* message gets saved in a persistent data store and the flow execution will terminate.

The second appearance of comments is related to message interspections, which mostly result in some form of variable setting. Take a look at the *getNodeValue* UML Call Operation Action. This action sets the variables *ProductionScheduleID* and *CallBackURI*. The first is a unique identifier for this schedule instance, which will be needed for the identification of the correct return message of the collaboration call (correlation). The second is necessary to realize a request-response interaction pattern by means of a call back at the end of the flow execution. Due to the use of an explicit return URL within the initiation message it is possible to send the response wherever appropriate.

Both comments have a link to a business rule constraint. Because different types of business rules can exist, a further classification is necessary. In this collaboration you see the BR classification, which means that the name of the node where the unique identifier can be found in the XML node tree shall be defined in an externalized business rule which allows to flexible set the *IDNodeName_Var* variable. This makes sense, because it is therefore possible to use the same collaboration for e.g. different language types of a structural identical *ProductionSchedule_Mes*. The XML distinguished node classification means that the business node name is not externalized but the node name is statically bound to the flow definition.

In addition, message types are referenced by name for each object flow. In the model organization those 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.

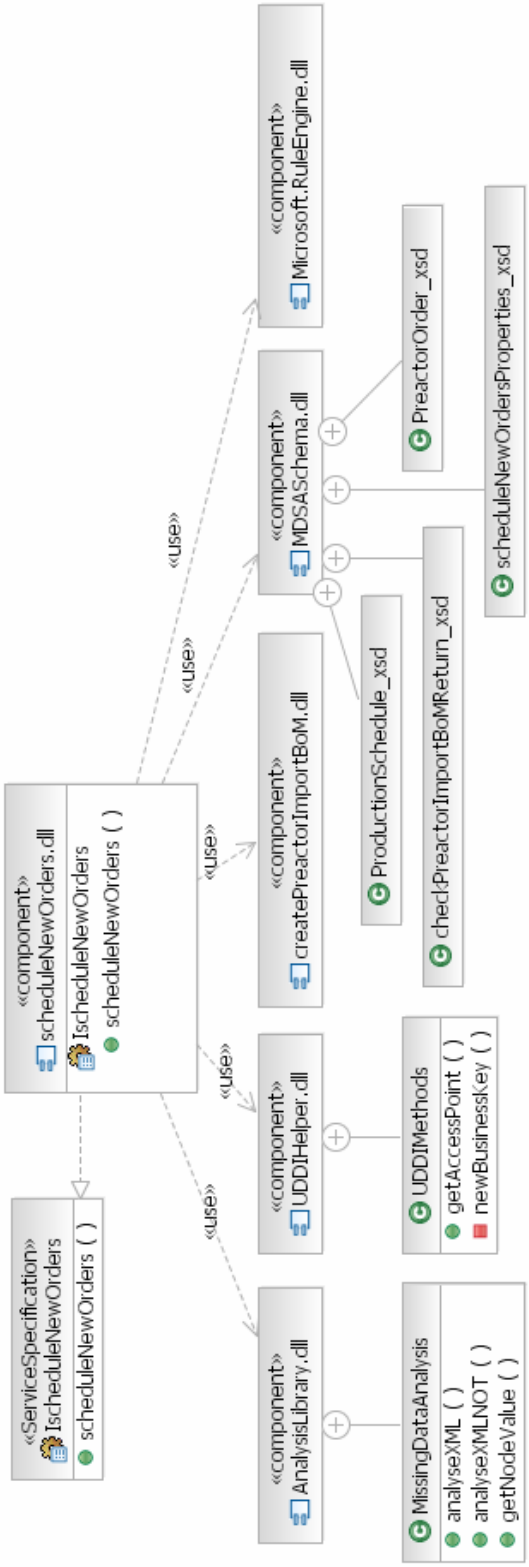


Figure 56. scheduleNewOrders implementation diagram of the Component View perspective

We also have to discuss shortly the *Component View* perspective, where the realizations of *Service Specifications* by means of components and classes are modeled. Figure 56 depicts the implementation respective realization of the *scheduleNewOrders Service Specification*.

This is the well known OO modeling world, thus it is expected that the reader understands this diagram without further explanation. What should be said is that this is the physical implementation of the interface and not the description of the collaborations in which it participates.

So if you compare Figure 54 with Figure 56 it should be obvious that in the latter the participating interfaces with SOAPDoc *Service Channel* bindings are missing, whereas the BTDirect bindings appear. For instance, included is the createPreactorImportBoM.dll assembly which realizes the participating *Service Collaboration.* Or the MDSASchema.dll component which owns the schema definitions for the data types referenced in this interface realization (*DataDefinition* role). The third BTDirect binding to the *Configuration* role can not be seen, because the *IProductionScheduleCheckConfiguration Service Specification* (see Figure 50 and read clause 7.4.2.2 for a further discussion of that issue) is not realized by means of a own component but it is implemented as an orchestration part of the scheduleNewOrders.dll component.

Although it can easily be done, a further decomposition of the internal structure of this component does not take place in the modeling environment because this is not the focus of the thesis. In addition, information redundancy would be the result. For OO decomposition we can switch to the development environment, which holds all the information from that point on. It was already said that the development environment is Visual Studio .NET 2003. With the installation of BizTalk Server 2004 the BizTalk Orchestration Designer and other tools are added.

The upper part of the *scheduleNewOrders Service Collaboration* was again chosen to present a short introduction into this final step of platform specific process modeling. Please compare Figure 57 with the platform independent process model in Figure 55 which utilizes UML as the *Model Representation Language*. In Figure 57 you see the platform specific *Composition Model*, and XLANG/s will be the *Executable Composition Language*, just as it was outlined in Figure 37.

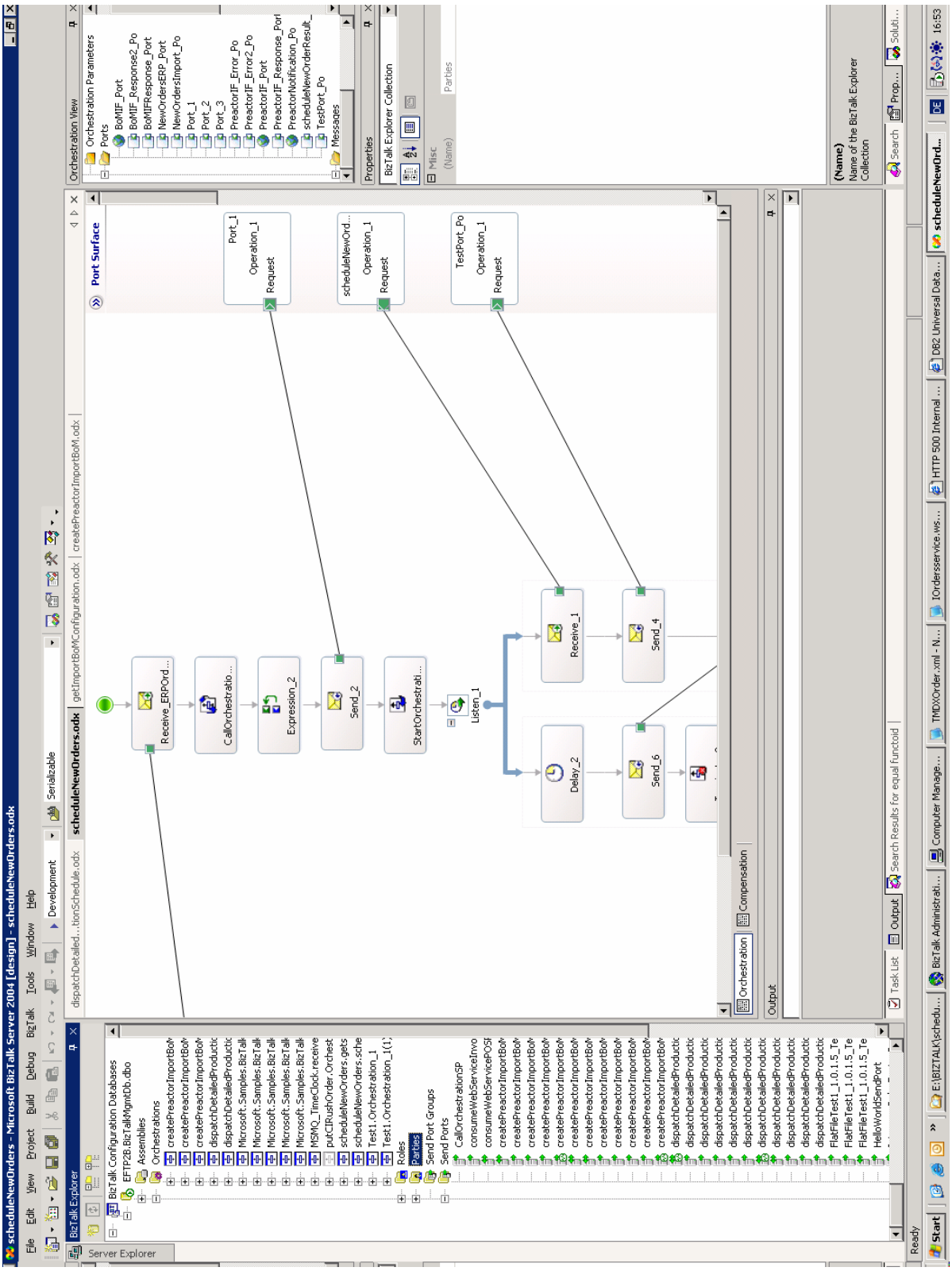

Figure 57. BizTalk 2004 Orchestration Designer view of the scheduleNewOrders collaboration

## 7.4. Production Operations Management Demo Scenario Service Providers

The *Service Providers* shall be described in greater detail, although they become obvious when we look at the *Detailed Production Scheduling* activity in Figure 50. Most of them were coded by the author, others were already available. We should remember that all *Service Providers* are logically grouped in *Service Partitions*, which represent the ANSI/ISA 95 activities.

Some general information about the service level shall be given here, although a detailed discussion about those matters will occur during the *Service Provider* discussion in the following.

There are three means to model usage dependencies at the service level. The first two representations are interchangeable, whereas the latter one is compulsory for collaboration definition.

*Service Provider* ß à *Service Specification*: The required interface compartment of the *Service Provider* (UML Component) gets populated by means of *Service Specifications* (UML Interfaces) (e.g. Figure 66)

*Service Provider* ß à *Service Provider*: In Figure 50 the Service View overview diagram was presented, a means to model usage dependencies between *Service Providers*.

*Service Specification* ß à *Service Specification*: The Collaboration composite structure diagram (Figure 54) defines dependencies at the interface level.

For the implementation of service compositions patterns were defined. Some of them are BizTalk specific, whereas others are means to standardize certain interaction patterns. Theses patterns will be discussed in subchapter 7.4.2.1.

### 7.4.1. Infrastructure Services

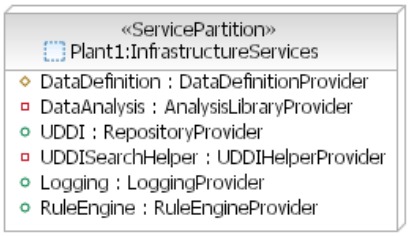


Figure 58. Infrastructure Service Partition

The infrastructure *Service Partition* is depicted in the figure above. Those service providers belong to this partition, which offer generic functionality within the architecture. The providers can be divided into three groups:

- Mandatory service providers (DataDefinitionProvider, RepositoryProvider, RuleEngineProvider): Their services are absolutely necessary for overall system operability. Without them the required system behavior can not be realized.
- Redundant service providers (AnalysisLibraryProvider, UDDIHelperProvider): Those services are not critical, but they provide functionality which positively influences system performance or flexibility.
- Optional service providers (LoggingProvider): For the basic system behavior the availability of those plays no role, but they are useful for requirements beyond process execution.

#### 7.4.1.1. Data Definition Provider

*Service*: DataDefinitionNET


Figure 59. Service Specification of DataDefinitionProvider

*Description*: This provider encapsulates all public data type definitions. Because the demo scenario is XML based, only schema definition files (.xsd) are affected. In a distributed system the at least virtual centralization of data definitions resulting in a single point of access can be the aim. This prevents different versions of the same type and increases maintainability. Although not implemented at the present stage, this concept would allow dynamic data binding too, which means that during run time each instance which handles the data object requests the actual data format before execution starts. A hybrid approach of caching combined with periodical definition update or update subscription would decrease network traffic. In our scenario those service providers which interact with the data definition provider are all deployed as BizTalk assemblies, thus this provider is compiled as a BizTalk project and a resulting .NET assembly (MDSASchema.dll). There are two situations in a BizTalk orchestration which require scheme referencing.

First the use in a message type property (e.g. Message Type = MDSASchema.ProductionSchedule) or second as a source or destination scheme in a BizTalk map (e.g. the dispatchDetailedProductionSchedule.forRelease_to_CIOrders map references MDSASchema.DetailedProductionSchedule as source scheme and MDSASchema.TMDXOrder as destination scheme.) The following schemes were defined as follows:

ProductionSchedule.xsd

*Description*: This is the order relevant information which is passed on from Level 4 (typically an ERP or some kind of PPS system) to Level 3.

*Structure*: As mentioned already, this scheme is not compliant with ANSI/ISA 95 *Production Schedule* definition. Nevertheless, the basic structure with *Production Requests* and finally *Production Request* is the same. The fields defined for the demo scenario is shown in Figure 60. It is assumed that the data source (ProductionScheduleProvider) has some rough production planning capabilities, setting start time- and due time-values for each order. Due to sequential planning of time slots and resources the quality of these values is insufficient, hence the use of a subsequent detailed scheduling system.

Beside those standard nodes two special nodes were added, the CallBackURI node and the ID node. The first allows for asynchronous interaction patterns with call-back functionality and the latter for unique identification of each released ProductionSchedule. Both nodes were promoted. CallBackURI is a distinguished field, which means that the value can be directly read in a BizTalk orchestration expression shape. The property field ID makes it possible to hold the node value in a property scheme node. This is necessary for correlation, a mechanism to identify the correct response message in asynchronous interactions by means of unique identifiers.

*Referenced by Service Specification*: IscheduleNewOrders (see Figure 56)

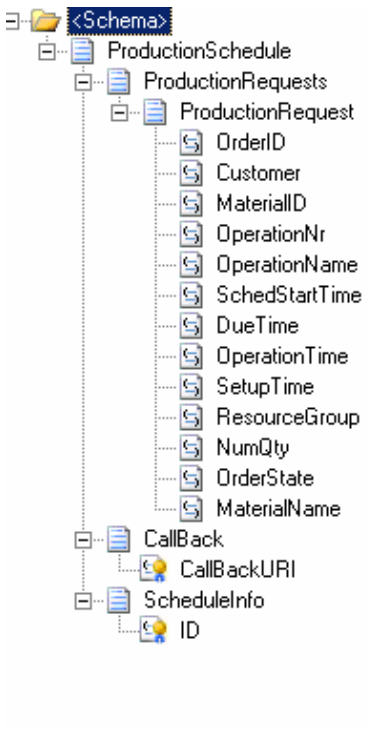


Figure 60. ProductionSchedule.xsd

scheduleNewOrdersProperties.xsd

*Description* and *Structure*: This schema has just one node, namely Property1. This is the node where the ProductionSchedule ID value gets stored during orchestration instance execution.

*Referenced by Service Specification*: IScheduleNewOrders (see Figure 56)

PreactorOrder.xsd

*Description*: This scheme represents the data type used to interface the Preactor scheduling software.

*Structure*: Figure 61 depicts the tree view of the scheme. The ERPWorkOrders node has an unbound "group maximum occurs" property, which means that the ERPWorkOrder node can occur more than once. As with the ProductionSchedule scheme, each operation constitutes an ERPWorkOrder.

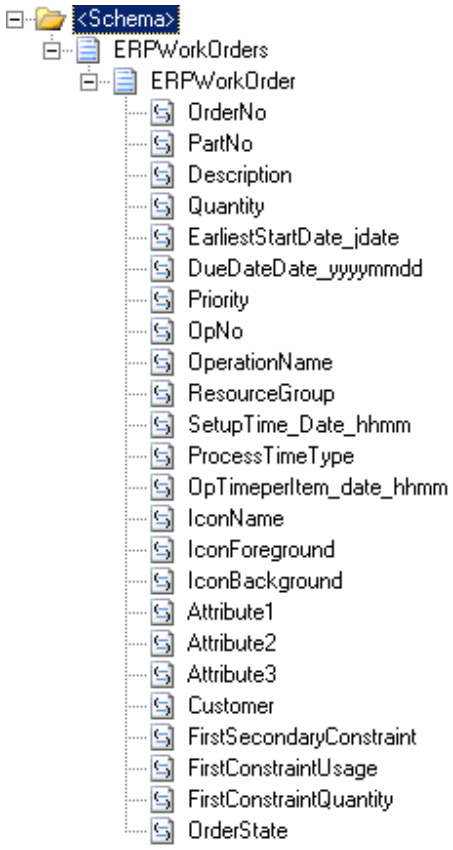


Figure 61. PreactorOrder.xsd

*Referenced by Service Specification*: IScheduleNewOrders (see Figure 56)

checkPreactorImportBoMReturn.xsd

*Description*: This scheme defines the payload of the return message of a createPreactorImportBoM service call.

*Structure*: The scheme comprises one *Return* node and two child field elements, *ReturnMessage* and *PackageID*. *ReturnMessage* is a distinguished field and holds a string value identifying the invocation result (value "1" for success). *PackageID* is a property field. Again, this property definition is necessary for correlation type definition. The *scheduleNewOrders* orchestration and the *createPreactorImportBoM* orchestration correlate on this property.

*Referenced by Service Specification*: IScheduleNewOrders (see Figure 56), IcreatePreactorImportBoM

NewPartNumbers.xsd

*Description*: This is the scheme used to build the return message from the IBoM.checkBoMEntryXML service operation.

*Structure*: When the ProductDefinitionProvider checks the *Production Schedule* and detects missing BoM entries for occurring *MaterialIDs*, than each missing part number is included as a *MissingPartNumber* field value. Figure 62 depicts the tree view of the scheme. The root child field *Missing* is a boolean value, indicating whether a BoM entry is missing at all. *FileType* returns the name of the data source which holds the BoM data.

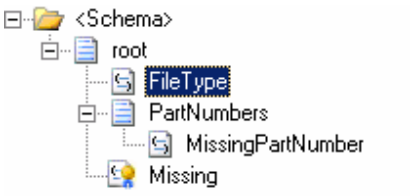


Figure 62. NewPartNumbers.xsd

*Referenced by Service Specification*: IcreatePreactorImportBoM, IBoM

DetailedProductionSchedule.xsd

*Description*: This scheme contains the basic information (time and primary resource) necessary for operation execution after detailed scheduling took place. In this version of the demo scenario secondary resource data is not passed from the scheduler to the dispatcher.

*Structure*: Like the other operation related schemes it is a two level structure, with *DetailedProductionRequests* as root element and one to many *DetailedProductionRequest* nodes. The SchedulingProvider sets the *Setup_Start*, *Start_Time* and *End_Time* values as well as the *Resource* value (the initial ERPWorkOrder only specified the *ResourceGroup)* What is more, information about partly finished operations (*QtyAcceptSum*, *QtyRejectSum*) is included, together with the *Operation_Progress* state.

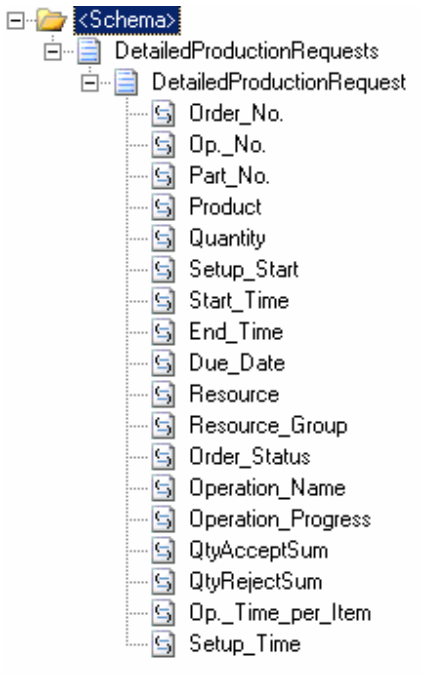


Figure 63. DetailedProductionSchedule.xsd

*Referenced by Service Specification*: IProductionDispatch, IPreactorIF (for Excel XML to DetailedProductionSchedule transformation)

TMDXOrder.xsd

*Description*: As we will describe in Section 7.4.5.1, one of the *Service Providers* responsible for *Production Execution Management* is the Cell Integrator Manager. It is capable of integrating NC-machines into a SOA, the CI_Manager component expects the TMDXOrder format.

*Structure*: If you compare a *DetailedProductionScheduleRequest* with a *TMDXOrder*, it should be obvious that, again, the abstraction level was lowered. At the machining level information about additional resources, in our case *ClampData* and *NCData* is needed, along with more information about the actual operation progress. This data has to be provided by subsequent activities, but these steps are not implemented in the demo scenario yet.

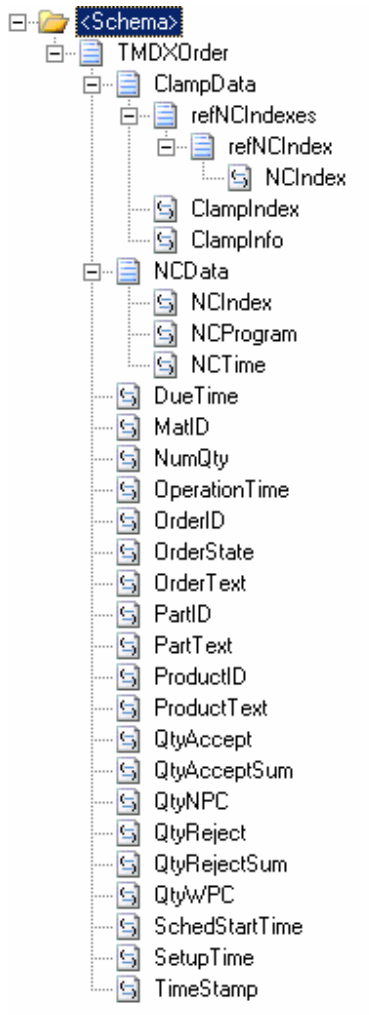


Figure 64. TMDXOrder.xsd

*Referenced by Service Specification*: IProductionDispatch, IOrders (the reference is hard coded in the application).

#### 7.4.1.2. Repository Provider

*Service*: RepositorySOAP


Figure 65. Service Specification of RepositoryProvider

*Operation*: All methods provided by the UDDI.API.InquireMessages class and the UDDI.API.PublishMessages class compiled in the uddi.api.dll assembly

*Description*: The Microsoft Windows Server 2003 inbuild UDDI Services (Version: 5.2.3790.0) realize the RepositoryProvider functionality. A detailed description of the offered functionality is beyond the scope of the theses, so at this point we will take a look at the fundamental features regarding this project.

- Access: UDDI Services can be accessed through a Web-based user interface or programmatically through a SOAP interface (Web Service). The UDDI API for .Net (Microsoft.Uddi.dll Version 2.0) eases the interaction coding. During design time of the scenario the user interface was the main window to insert all the relevant information. Programmatically the SOAP interface was used in conjunction with the UDDIHelper façade introduced below.
- Providers: It was decided to map each *Service Partition* to a UDDI service provider and every *Service Specification* operation to a UDDI service due to the small number of operations in addition to the demand for single operation visibility in the UDDI. This means that MDSA *Service Providers* and *Services* are not present in the UDDI. One of the main reasons for this was the fact that dynamic binding port (access point retrieval during run time) incorporation is more easily feasible if the UDDI binding search request directly returns the right operation call URL. In addition, each published BizTalk orchestration is deployed as a Web Service with a single operation.
- Instance Infos: The so called tModels are the means for detailed specification of UDDI entities, especially for bindings. Thus it is possible to publish technical information for an interface, such as parameters or a WSDL file. But in general tModels can point to every kind of overview document, not just WSDL files. Moreover, tModels can be further described by categorizations. Every SOAP binding in the UDDI has a added tModel (e.g. wsdl:IF:BoM) pointing to the corresponding WSDL file. Notice that the deployment information (the service and port tags) was removed from the WSDL files to create a pure Web Service interface description. Runtime search for access points is therefore enabled. Moreover, each binding instance info tModel is marked with the key value *wsdlSpec* of the uddi-org:types categorization schema. The latter extension allows for programmatically UDDI search of WS specifications, a service provided by Visual Studio when adding a Web reference. The second type of tModels is related to QoS issues. STATUS:production (:temporary, :test, :unavailable) is an example for a first, rough classification of bindings. The third type gets used for categorization.
- Categorization: Categorization schemes can be imported into UDDI Services. A Microsoft Tool called UDDI Categorization Scheme Editor (Version 5.2) allows for XML scheme creation with automatic tModel key and value management. The following categorizations are actually deployed as tModels:

Ø CAT:ISA S95 Activity Model (a group of subcategories representing the ANSI / ISA 95 activities for Production Operations Management)

- Ø CAT:ISA S95 Equipment Hierarchy (a tree structure representing the ANSI/ISA 95 Hierarchy Model)

For a detailed discussion about the hands on experiences with the UDDI Services application as well as the integration into the MDSA for the Shop Floor take a look at the corresponding *Service Provider* description. The following figure depicts a screenshot of the UDDI publish frame, illustrating the providers, a service (=operation) of Cell_1 and the binding including the linked WSDL tModel which is further specified (Overview Document) in the main window.

*Referenced by Service Specification:* IUDDIHelper

#### 7.4.1.3. UDDIHelperProvider

*Service*: UDDIHelper


Figure 66. Service Specification of UDDIHelperProvider

*Description*: The *Service Specification* and the operations carried out by this provider are shown in Figure 66. Above a required interface compartment added to the provided interface compartment. This is a way to express usage dependencies between this service provider and the SOA interfaces (*Service Specification*). The purpose of the service provided is to encapsulate some of the Repository Provider service operations and offer coarse grained granularity. Thus a number of repository interactions in order to add a binding (not knowing whether the service still exists) can be combined in one single request response pattern. What makes this concept attractive is the possibility to incorporate control logic as well as failure handling in the code. Due to the limited number of service consumers this service was realized as a .NET assembly and not as a Web Service. The full potential of this approach was leveraged in the BizTalk orchestrations, where the service calls were included in expression shapes, avoiding failure handling within the service flow and increasing performance. An important consideration is to make the implementation class (UDDIMethods) serializable, otherwise a BizTalk reference is not allowed.

Next to the dependent *Service Specifications* mentioned below this *Service* is also referenced by the SchedulINA application.

*Referenced by Service Specification:* IscheduleNewOrders, IcreatePreactorImportBoM


Figure 67.UDDI Services user web interface (administrator role)

#### 7.4.1.4. AnalysisLibraryProvider

*Service*: AnalysisLibrary

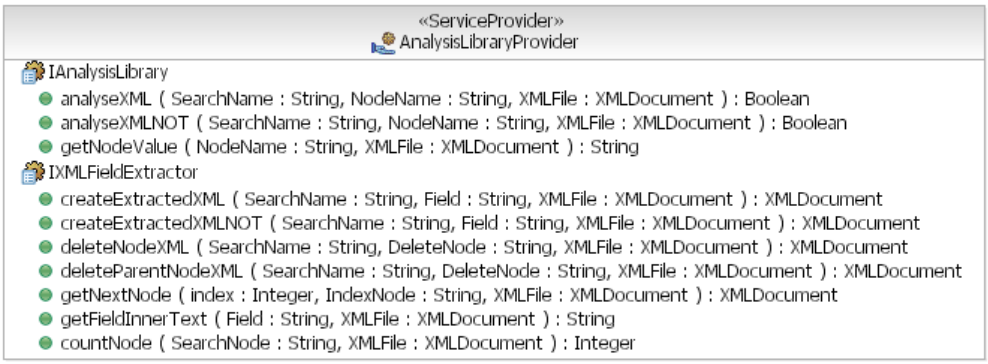


Figure 68. Service Specification of AnalysisLibraryProvider

*Description*: The *Service Specification* and the operations of this provider are shown in the figure above. The purpose of the service provided is to perform some XML file analysis and return either boolean values, node text or transformed XML files. These *Services* are referenced in the BizTalk assemblies, where service calls are included in orchestration expression shapes. The return values of the type boolean are used to determine the orchestration control flow by means of decision points. The author is aware of the fact that the examination of XML files could be realized by xpath expressions as well, avoiding the external dependency. And instead of the methods for XML file manipulation BizTalk transform shapes could have been used. But transform shapes are proprietary MS constructions, hampering the mapping of the flow definition to other execution languages like BPEL4WS. More generally, the aim was to make as much flow logic external as possible, thus leveraging functionality reusability. In addition, in some cases it is easier and faster to rather code the methods than to define the xpath expressions, especially when you want to include variables which are populated during runtime by means of business rule calls. Due to the limited number of service consumers this service was also realized as a .NET assembly and not as a Web Service. It is important to make the implementation classes (XMLFieldExtractor, MissingDataAnalysis) serializable, otherwise a BizTalk reference is not allowed.

*Referenced by Service Specification:* IcreatePreactorImportBoM, IscheduleNewOrders (IAnalysisLibrary only), IdispatchDetailedProductionSchedule (IXMLFieldExtractor only)

#### 7.4.1.5. LoggingProvider

*Service*: BLogServer


Figure 69. Service Specification of LoggingProvider

*Description*: This service represents the central hub for log messages in the whole architecture, thus the operation CreateBLOGEntry offers a broad set of optional parameters. So far, only the ProductionResourceProvider uses this service by means of *CTStamp* (time stamp of the message) and *BMessage* (message text) parameters. The LoggingProvider is deployed as a stand-alone web component.

*Referenced by Service Specification:* IOrders

#### 7.4.1.6. RuleEngineProvider

*Service*: RuleEngine

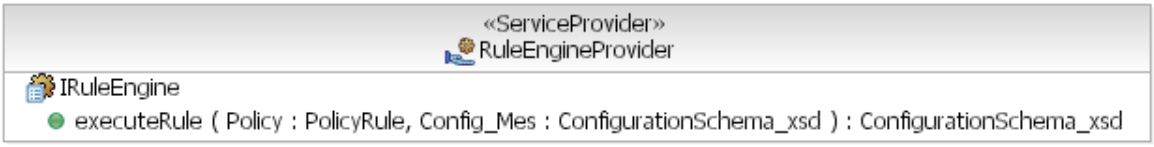


Figure 70. Service Specification of RuleEngineProvider

*Description*: The purpose of business rules regarding BPM was described in subchapter 4.1.3.3. Microsoft Business Rule Composer 3.0 is the rule engine in this architecture. Thus, there is a tight coupling between CallRules shapes in the BizTalk orchestrations and the rule execution. Nevertheless, it is necessary to make this interaction externally visible. Rules are logically grouped in policies. Each rule is dependent on data sources, which are in our case XML schemas (databases and .NET classes are alternative data sources). The MS tool allows the definition of so called vocabularies, an abstraction layer between data source and rule definition. Vocabularies contain reusable mappings between user-friendly text and the underlying data sources used in a rule definition. When a rule is executed, the scheme nodes are populated dependent on the rule logic and an instance of the schema is returned to the caller. Therefore for an operation call the name of the policy and an instance of the scheme are the input parameters. Figure 71 depicts a screenshot of the Business Rule Composer. The condition for the rule configuration, which follows an IF – THEN – ELSE pattern, was set “1 is equal to 1”, resulting in a rule execution every time the rule gets invoked. The values defined in the actions and described by means of the

vocabulary (Facts Explorer window) are the same as in a typical config file, e.g. the UDDI inquire URL or some node names of messages which are extracted in the orchestration. This allows for flexible changes of message formats without changing the service flow definition. Hence versioning has to be supported. The Facts Explorer also contains the data source references. Some of the binding information like document type and physical scheme location can be seen in the Properties window in Figure 71.

*Referenced by Service Specification:* IcreatePreactorImportBoM, IscheduleNewOrders, IdispatchDetailedProductionSchedule

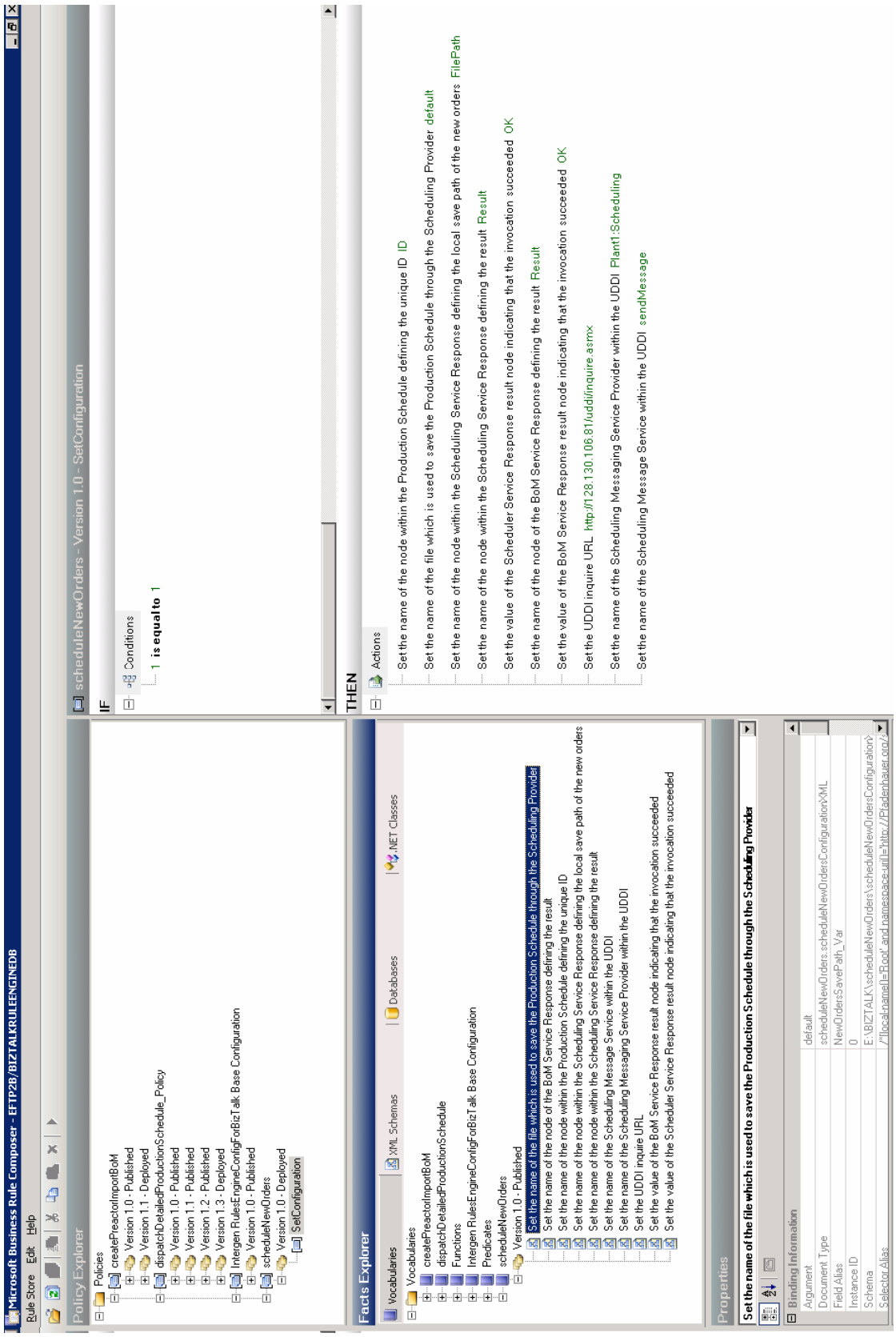


Figure 71. Microsoft Business Rule Composer screenshot

## 7.4.2. Detailed Production Scheduling Services

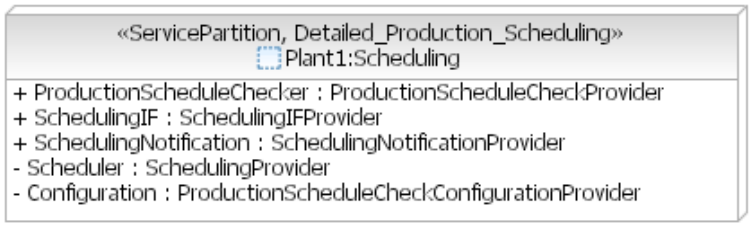


Figure 72. Plant1:Scheduling Partition (Detailed Production Scheduling)

### 7.4.2.1. ProductionScheduleCheckProvider

*Service*:

- scheduleNewOrdersFSO (this is the file system object activation port)
- scheduleNewOrdersSOAP (this is the WS activation port)
- createPreactorImportBoMBT (this is the port which can only be accessed by internal BizTalk Orchestration Calls)
- createPreactorImportBoMWS (this is the WS activation port)


Figure 73. Service Specification of ProductionScheduleCheckProvider

*Description*: Both interfaces are complex services implemented as BizTalk orchestrations. Both expect the ProductionSchedule as an import parameter. Due to the BizTalk Web Services Publishing Wizard it is rather easy to map FSO ports to WS ports running under MS IIS. createPreactorImportBoM is initially an orchestration without a Receive shape whose activation property is set to true. This is necessary for internal BizTalk calls, but makes it impossible to instantiate the orchestration externally. To achieve this, a "mini" orchestration has to be deployed, consisting of a port (FSO port which becomes a WS port after publishing), a receive action (activation property set to true) and an asynchronous Orchestration Start shape which initializes the main control flow.

Hence this construction shall be called a proprietary BizTalk Activation Pattern. Because all external orchestration calls are asynchronous, the incoming messages include an optional call back URI field.

This leads to a common CallBack Pattern realized in createPreactorImportBoM, namely the extraction of a call back address at the beginning of the control flow. The forwarding of the return message at the end of the control flow (decision either to a standard FSO port or a dynamic binding WS port binding to the extracted address) depends on whether an address was found in the call back URI field. Whereas internal activations are either asynchronous or synchronous, both are possible.

The design process for the scheduleNewOrders service was described in detail in the previous sections. As a full coverage of the whole flow was not possible, a short functionality description shall be given. After the flow is initialized by means of an arriving *Production Schedule*, a synchronous orchestration call takes place to activate a separated orchestration whose purpose is to perform a business rule call.

This Configuration Pattern was realized for all BizTalk orchestrations and makes the rule call reusable. Moreover, the configuration orchestration itself can be provided as a template. The getscheduleNewOrdersConfiguration orchestration has an outgoing Config_Mes message which returns the business rule values to the calling orchestration. After the configurable variables are set, CallBackURI_Var (extinguished field) and PackageID_Var (AnalysisLibrary *Service* call) get extracted from the message. Messages are periodically sent to FSO ports which serve as persistent storages for messages, because in the BizTalk database they are transient. Then an asynchronous orchestration call to the createPreactorImportBoM operation takes place, the result message is polled from a FSO port.

This Service Call Pattern uses a delay shape to set a timeout and terminates if no return message arrives at the polling port. The execution only continuous if the called orchestration returned an OK value. The ProductionSchedule gets transformed to a PreactorOrder and is send to the SchedulingIFProvider. For this request-response interaction the above pattern is applied too, terminating the flow after a certain timeout. If the response message contains a certain value (set in the Configuration Business Rule), the flow calls the BoMSOAP service of the ProductDefinitionProvider to provoke the creation of a suitable BoM file for import into the scheduling application Preactor. If this was successful, all customers subscribed in the UDDI receive a message via MSMQ saying that new orders are waiting for import. The new orders are included in the message body.

This Notification Pattern was rather complex to implement, because it is not known whether and how many bindings are registered. For that reason the UDDIHelper service providing index based UDDI address search capability is used. Guaranteed message delivery is not achieved due to a one way “fire and forget” broadcasting mechanism. Otherwise some recovery concept would be needed if one receiver does not respond. The use of a timeout raises the question of locking the UDDI for the whole transaction to keep the actual state consistent.

The createPreactorImportBoM service calls the ProductionDefinitionProvider to check whether all parts in the ProductionSchedule have a BoM entry available. This check has to be performed twofold: for possibly included sales orders and for the rest of the ProductionRequests. This is necessary because the sales order BoMs and part BoMs are hold in different data tables. The incorporation of sales orders in detailed

scheduling tasks is unusual, but with a sales order BoM it is possible to keep the information which *Production Requests* belong to one and the same sales order. Both checks are provided by a single IBoM operation. If sales order entries or part entries are missing in the tables, a message including the missing numbers is sent to the ProductDefinitionProvider. The check result is indicated to the service consumer by means of a return value of 1 if no entry was missing.

*Referenced by Service Specification:* IReleasedProductionSchedule

#### 7.4.2.2. ProductionScheduleCheckConfigurationProvider

*Service*:

- ProductionScheduleCheckConfigurationBT (this is a port which can only be accessed by internal BizTalk Orchestration Calls)

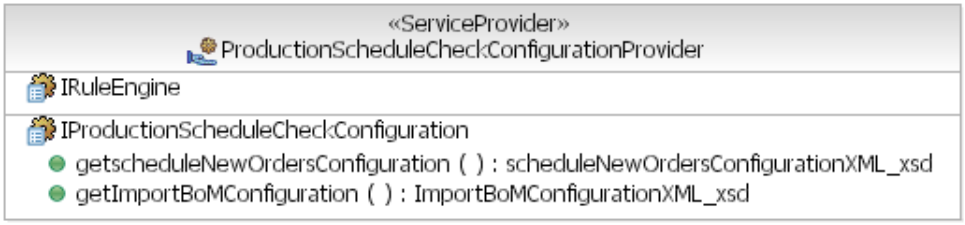


Figure 74. Service Specification of ProductionScheduleCheckConfigurationProvider

*Description*: A Configuration Pattern was realized for all BizTalk orchestrations and makes the rule call reusable. This means that each rule call is wrapped in an orchestration, which contains a message construction activity and an internal IRuleEngine call packed in a scope. Moreover, this configuration orchestration itself can be provided as a template. The getscheduleNewOrdersConfiguration orchestration has an outgoing Config_Mes message of type schedulNewOrdersConfigurationXML which returns the business rule values to the calling orchestration. Although we model the two configuration operations for the ProductionScheduleCheckProvider as part of a single *Service Provider*, the implementation of each configuration operation as a BizTalk orchestration was deployed together with the ProductionScheduleCheckProvider orchestrations. This matter of fact can be modelled in the *Component View*. Hence in the implementation diagram for scheduleNewOrders.dll (Figure 56) all configuration interactions are hidden as internal activities within the assembly and a direct dependency relationship with the Microsoft.RuleEngine.dll is shown. This deployment decision was made because in this demo scenario caller, configuration provider and execution engine operate in the same environment. If that would not have been the case, the methodical separation between main orchestration, configuration orchestration and execution process through distinctive providers would have been necessary.

*Referenced by Service Specification:* IscheduleNewOrders, IcreatePreactorImportBoM

#### 7.4.2.3. SchedulingIFProvider

*Service*:

- PreactorIFSOAP (WS port)

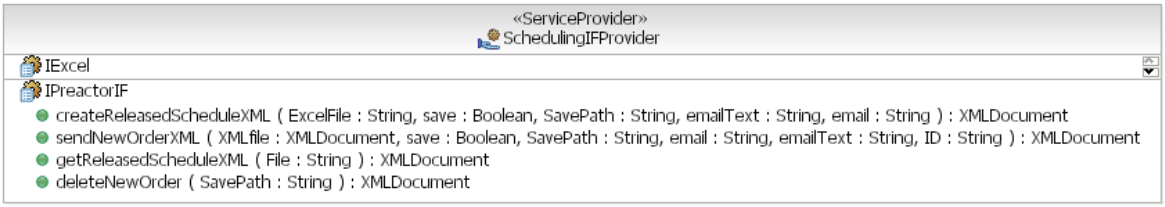


Figure 75. Service Specification of SchedulingIFProvider

*Description*: This *Service Provider* is a wrapper program for the scheduling software Preactor 9.2. Preactor is an APS (Advanced Planning and Scheduling) system for detailed production scheduling. Beside the standard functions such as APS scheduling rules combined with a huge number of planning board facilities (drag and drop, capacity graphs, gantt chart, etc.) Preactor can be integrated into a Client-Server architecture via ActiveX (Microsoft COM technology). This means that all Preactor objects are accessible through ActiveX objects. As a server it offers full access to the database and the scheduler functionality via the Open Planning Board object and its methods library. So the user written Client/Server code is used to interact with the software or react to human planner activities offering additional functionality. Unfortunately, the present release does not allow using ActiveX objects within the ASP.NET environment, although this would be necessary to wrap Preactor method calls as web methods and provide them as Web Services running under APS.NET. Thus the decision was made to use the basic import/export functionality for data via comma separated files and provide services which do the .xml - .csv transformation. This is basically what the IPreactorIF operations do. To achieve this, the Excel library is used for the mapping.

sendNewOrderXML is the main operation to send the XML orders in a free format, although the use of PreactorOrder.xsd (see above) is expected for the import into Preactor via the proper script. This script transforms the NewOrdersTest[…].csv (e.g. NewOrdersTest for ID 111 at 03.05.2006_14_14_18; ID is the ID input parameter) file and maps the columns to the internal database table fields. The script also merges the order and the ImportBoM[…].csv (e.g. ImportBoM for ID 111 at 03.05.2006_14_14_20) file to create the complete product structure. The script is predefined for the PreactorOrder format. But this architecture is very flexible insofar as the manual script import plus the prepared .csv file allows for different sources with different formats. The flexibility of this approach has been increased further by the implementation of a web.config configuration file, where the settings for file name and location can be made.

createReleasedScheduleXML transforms a file format (location specification via parameter or ExcelFile="default"/SavePath="default" for web.config values)

accepted by Excel into a XML file. Although a generic functionality, it is used to map the Preactor output .csv files to .xml files.

getReleasedScheduleXML returns the specified or web.config default (value "default" for the File string parameter) XML file to the caller.

deleteNewOrder deletes a file (location specification via parameter or SavePath="default" for web.config values)

*Referenced by Service Specification:* IscheduleNewOrders, HMI Application SchedulINA

#### 7.4.2.4. SchedulingNotificationProvider

*Service*:

- SchedulingNotificationSOAP (WS port)

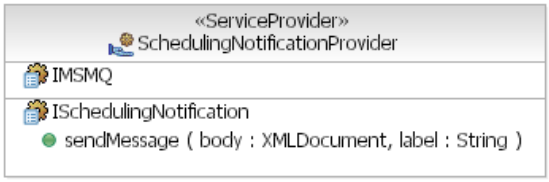


Figure 76. Service Specification of ScheduleNotificationProvider

*Description*: This very simple service wraps the Microsoft Message Queuing application and allows the submission of new messages. MSMQ is a standard application of every Windows operating system and has to be activated at the host computer. The sendMessage operation is a one way SOAP interaction, thus there is no return value. If the *label* parameter equals the *ProductionScheduleLabel* config value then the message is routed in the queue which is defined by the *ProductionScheduleQueuePath* config value. Otherwise the *GeneralQueue* config value sets the target queue which has to be made available in MSMQ.

*Referenced by Service Specification:* IscheduleNewOrders

#### 7.4.2.5. SchedulingProvider

*Service*:

- SchedulingNotificationSOAP (WS port)


Figure 77. Service Specification of SchedulingProvider

*Description*: The SchedulingProvider is a role carried out by an application (Preactor 9.2) and a human. The aim is to support the scheduling worker as much as possible and provide all information necessary while leveraging the human knowledge as good as possible. This is feasible because fully automated scheduling systems without human interaction proved to result in sub-optimal schedules, especially when it comes to the issue of rescheduling which demands for reactive systems. A human worker / application / supporting functionality approach seems to be very promising. The supporting functionality concerning data import is related to the fact that all files necessary for import are ready in the defined folder due to the SOA concept (SchedulingIFProvider, ProductDefinitionProvider support). Supporting functionality is also gained through the use of the HMI application SchedulINA (Section 7.5).

Production Resource Management for primary resource master data is part of the scheduling software and yet not accessible through public services. Hence it is not included in the model.

### 7.4.3. Product Definition Management Services

> *Product definition management shall be defined as the collection of activities that manage all of the Level 3 information about the product required for manufacturing, including the product production rules.*
>
> *Product definition information is shared between product production rules, bill of material and bill of resources. The product production rules contain the information used to instruct a manufacturing operation how to produce a product.*
>
> *ANSI/ISA (2005)*

In the given scenario two roles are assigned to the *Product Definition Management* activity, ProductDefinition and NC_Comm. Remember Figure 52, which introduced the concept of partitions and their relations to roles and *Service Providers* as role types. These roles are implemented by two *Service Providers*, ProductDefinitionProvider and NC_CommProvider. Both *Service Providers* are implemented as Web Services and use SOAP over HTTP as transport mechanism.

#### 7.4.3.1. ProductDefinitionProvider

*Service*:

- BoMSOAP (WS port)

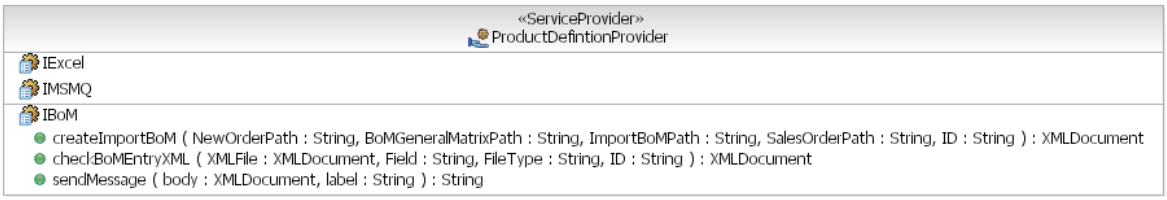


Figure 78. Service Specification of ProductDefinitionProvider

*Description*: This is the interface to the product definition, more precisely to the BoM (Bill of Material) structure of the products. In the demo scenario only the BoM has relevance for *Production Schedule* examination. Only products for which a BoM definition exists are allowed. In the present situation the BoM data is distributed between Excel files (one for sales orders and BoMGeneralMatrix for products), a solution sufficient for our requirements. Keep in mind that in SOA the implementation of services is hidden from the requestor, thus from an external point of view it makes no difference whether Excel files or database tables constitute the data layer. Because Web Service engineering is not a core research objective, the coding was kept as simple as possible (e.g. no three-tire-architecture was implemented).

checkBoMEntryXML does the comparison between the *XMLFile* input XMLDocument and the Excel files in terms of *Field* value existence in both of them. The XML return document contains the *ID*, the *FileType,* a *Missing* field indicating if something is missing at all and, if yes, it also contains the missing part numbers. The *FileType* parameter has to be of type string and value *BoMGeneralMatrix*, *SalesOrder* or *ImportBoM* and indicates the data source which shall be used for comparison. The *Field* parameter is used to determine which node of the input XML file contains the search values. The default value derived from the present ProductionSchedule.xsd is *MaterialID*.

createImportBoM is the function which creates a comma separated file which features the BoM structure of every ProductionRequest identified by means of a unique ID. This means that the anonymous BoM structure in the BoMGeneralMatrix or the SalesOrder Excel file which contain only part IDs but no order IDs have to be extended by means of OrderID values derived from the new order Excel file created by the SchedulingIFProvider.

sendMessage is similar to the SchedulingNotification service in that it pushes messages into a MSMQ queue but follows the request-response interaction pattern.

*Referenced by Service Specification:* IscheduleNewOrders, IcreatePreactorImportBoM, HMI application SchedulINA

#### 7.4.3.2. NC_CommProvider

*Service*:

- NC_CommSOAP (WS port)

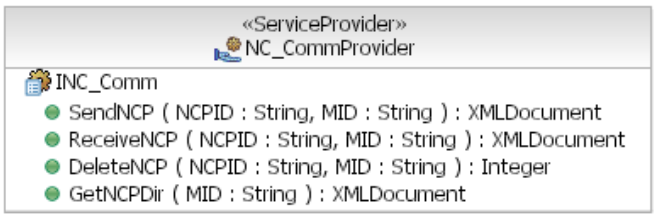


Figure 79. Service Specification of NC_CommProvider

*Description*: NC_Comm is the module which dispatches NC programs at the plant level and interacts with the appropriate ProductionResourceProviders. It is responsible for NC program transfer to and from the machines as well as the NC program management.

- Transfer from NC_Comm to the caller (SendNCP)
- Transfer from the caller to NC_Comm (ReceiveNCP)
- Deletion within NC_Comm (DeleteNCP).
- The GetNCPDir operation returns a machine specific list of available NC programs. NCPID is the NC program ID and MID the machine ID.

*Referenced by Service Specification:* There are NC_Comm modules which communicate directly with the machine control units, CAD/CAM applications or further NC Program sources.

## 7.4.4. Production Dispatching Services

> *Production dispatching shall be defined as the collection of activities that manage the flow of production by dispatching production to equipment and personnel. This may involve:*
>
> *a) Scheduling batches to start in a batch control system.*
>
> *b) Scheduling production runs to start in production lines.*
>
> *c) Specifying standard operating condition targets in production units.*
>
> *d) Sending work orders to work centers.*
>
> *e) Issuing work orders for manual operations.*
>
> *ANSI/ISA (2005)*

### 7.4.4.1. ProductionDispatchProvider

*Service*:

- ProductionDispatchFSO (File port)


Figure 80. Service Specification of ProductionDispatchProvider

*Description*: The dispatchDetailedProductionSchedule operation is a BizTalk orchestration which performs a registry lookup for the resource IOrders interface end points and dispatches the single TMDXOrder documents to the correct resource by

means of an IOrders.addOrder operation call. The BizTalk Configuration Pattern described in Section 7.4.2.1 was again used to implement the configuration rule setting.

*Referenced by Service Specification:* IScheduling

#### 7.4.4.2. ProductionDispatchConfigurationProvider

*Service*:

- ProductionDispatchConfigurationBT (this is a port which can only be accessed by internal BizTalk Orchestration Calls)

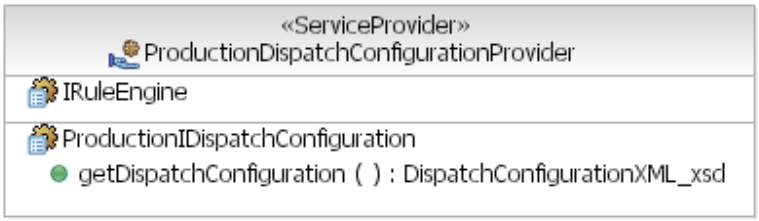


Figure 81. Service Specification of ProductionDispatchConfigurationProvider

*Description*: A Configuration Pattern was realized for all BizTalk orchestrations and makes the rule call reusable. See Section 7.4.2.1 for a description of this pattern.

*Referenced by Service Specification:* IProductionDispatch

### 7.4.5. Production Execution Management Services

> *Production execution management shall be defined as the collection of activities that direct the performance of work, as specified by the contents of the production dispatch list elements. The production execution management activity includes selecting, starting and moving those units of work (for example lots, sublots, or batches) through the appropriate sequence of operations to physically produce the product. The actual work (manual or automatic) is part of the Level 2 functions.*
>
> *NOTE — The definition of a sequence may take the form of a detailed production route specific for a particular produced item. Production execution transacts the individual units of work from one operation or step to the next, collecting and accounting for such things as actual materials consumed, labor hours used, yields and scrap at each step or operation. This provides visibility into the status and location of each lot or unit of work or production order at any moment in the plant, and offers a way to provide external customers with visibility into the status of an order in the plant.*
>
> *Production execution management may use information from previous production runs, captured in production tracking, in order to perform local optimizations and increase efficiencies.*
>
> *ANSI/ISA (2005)*

#### 7.4.5.1. ProductionResourceProvider

*Service*:

- OrdersSOAP (WS port)


Figure 82. Service Specification of ProductionResourceProvider

*Description*: This Service Provider was realized as a module of a stand alone Web Server by Ritter (2003) called Cell Integrator Manager. This application was developed as an interface to NC machines with machine order and NC program management as the two main tasks. For our demo scenario only the order, time slices and logging functionalities are relevant so far. The time slices that are dispatched by the application to reflect the order status changes are used to start the information flow back by means of the *Production Data Collection* activity. Thus the Cell Integrator without the INCPrograms *Service Specification* as it is displayed here can be used as a generic resource wrapper, for instance for hand operated equipment.

We simulated this by means of a .NET wrapper service which realizes just the IOrders interface. Another reason for this wrapping was the fact that the important changeOrderState operation returns originally a string-array type which is not allowed in .NET.

*Referenced by Service Specification:* IProductionDispatch, Cell Integrator GUI

#### 7.4.5.2. CellGUIProvider

This provider is mentioned here because it marks the low-level endpoint for order information management in our scenario. This user interface displays the resource specific orders and the user can enter confirmation data which leads to an IOrders.changeOrderState operation call. These events will create time slices which start the information flow upwards. For a detailed discussion of this application see Ritter (2003).

## 7.4.6. Production Data Collection Service

> *Production data collection shall be defined as the collection of activities that gather, compile and manage production data for specific work processes or specific production requests. Manufacturing control systems generally deal with process information such as quantities (weight, units, etc.) and associated properties (rates, temperatures, etc.) and with equipment information such as controller, sensor, and actuator statuses. The managed data may include sensor readings, equipment states, event data, operator-entered data, transaction data, operator actions, messages, calculation results from models, and other data of importance in the making of a product. The data collection is inherently time or event based, with time or event data added to give context to the collected information.*
>
> *ANSI/ISA (2005)*

### 7.4.6.1. TS_AnalyzerProvider

*Service*:

- TS_ReceiverSOAP (WS port)

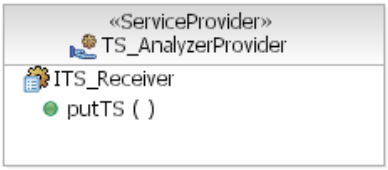


Figure 83. Service Specification of TS_AnalyzerProvider

*Description*: This is the service which receives the time slices from the production resources, for instance the Cell Integrator Manager. A single operation putTS is sufficient, which accepts parameters like *TimeStart*, *TimEnd*, *QuantityAccept* or *OrderID*. A detailed parameter discussion can be found in Ritter (2003), who defined the requirements for the TS_Analyzer provider when he developed the time slice export functionality of the Cell Integration application. Nevertheless, the provider was implemented for the first time by the author of this work.

*Referenced by Service Specification:* Cell Integration Manager

## 7.5. HMI Application SchedulINA

### 7.5.1. Introduction

In the previous section the Cell Integrator GUI was briefly described. As it was emphasized in the MDSA methodology discussion, this approach fully integrates human actions and out of it resulting demands. The example introduced there was the human scheduler role, which utilizes the Preactor 9.2 software with its SOA interface for service providing (e.g. updateDetailedProducitonSchedule) within the architecture. So far the integration into the SOA was restricted to import file preparation and message sending into a MSMQ queue. Rudimentary script import is possible in Preactor.

The given file import support functionality is not enough for the human scheduler to provide her or his services. Additional requirements are:

- Messaging: The user must have the possibility to check the received messages in the MSMQ queues. An extended version should include message sending to receivers listed in the registry.
- Registry Subscription: The GUI has to provide registry subscription functionality.
- Preactor Import/Export: Import/export script modification and activation shall be provided. To achieve this, the Preactor ActiveX functionality must be used.
- Additional SOA functionality: It must be possible to include WS service clients when necessary. In the first version a client for the BoM checking functionality shall be provided.
- File Explorer: Because the user has to work with import and export files, a FSO explorer is necessary.

The application in mind shall be highly modular in such a way as to enable customization according to the user requirements and rights. For instance the messaging functionality is crucial for every human participant in the SOA, whereas the Preactor modul is only interesting for the scheduler. It is important to understand that every user has to publish his or her message end points in the registry, the central hub for every system element. In addition, further subscriptions with dynamic end points must be possible.

In principle this could be done by means of the web GUI for the registry as well. This solution is more complex for the user, because he has to correctly enter the personal data such as service provider, service or binding details. Otherwise the entry would be useless. Secondly, the UDDI requires a windows authentication login. After user

account login the explorer shows only the entries published by this specific user. That is to say, after the first login the user sees an empty registry, thus all tModels and therefore classifications are missing. If he has at least coordinator rights he can switch the account, but then user rights management is nearly impossible.

Even if the user has all the information for consistent registry publications by hand it remains doubtful whether every user can handle this process. The GUI application presented in the following shall demonstrate how to hide this complexity from the user.

### 7.5.2. Implemented SchedulINA Functionality

The application fulfilling these requirements is named SchedulINA (Scheduling Interface and Notification Application) because it is a GUI for the scheduler role in the demo scenario. SchedulINA is a Windows application, written in Visual Basic in the Visual Studio .NET 2003 environment. It contains the typical Windows menu structure as well as windows cascade and tile functionality.

One major functionality block is related to messaging, providing a user interface to the MSMQ application as well as to the UDDI registry. Figure 84 depicts the user interface with the three main messaging windows open.

The *Domain Messaging 1* window shows the existing messaging queues at the local MSMQ instance, other instances are not available in the domain at the moment. The user has chosen to monitor the production schedule message related queue. If a new message arrives, a notification window will pop up, either in the SchedulINA or in the Preactor environment. The last option is useful if the user is not monitoring the SchedulINA application permanently but is focusing more on the Preactor software. Next to receiving and analyzing message bodies it is possible to send messages as well. The user can create local queues, a functionality which allows her or him to setup the application according to the specific role very quickly.

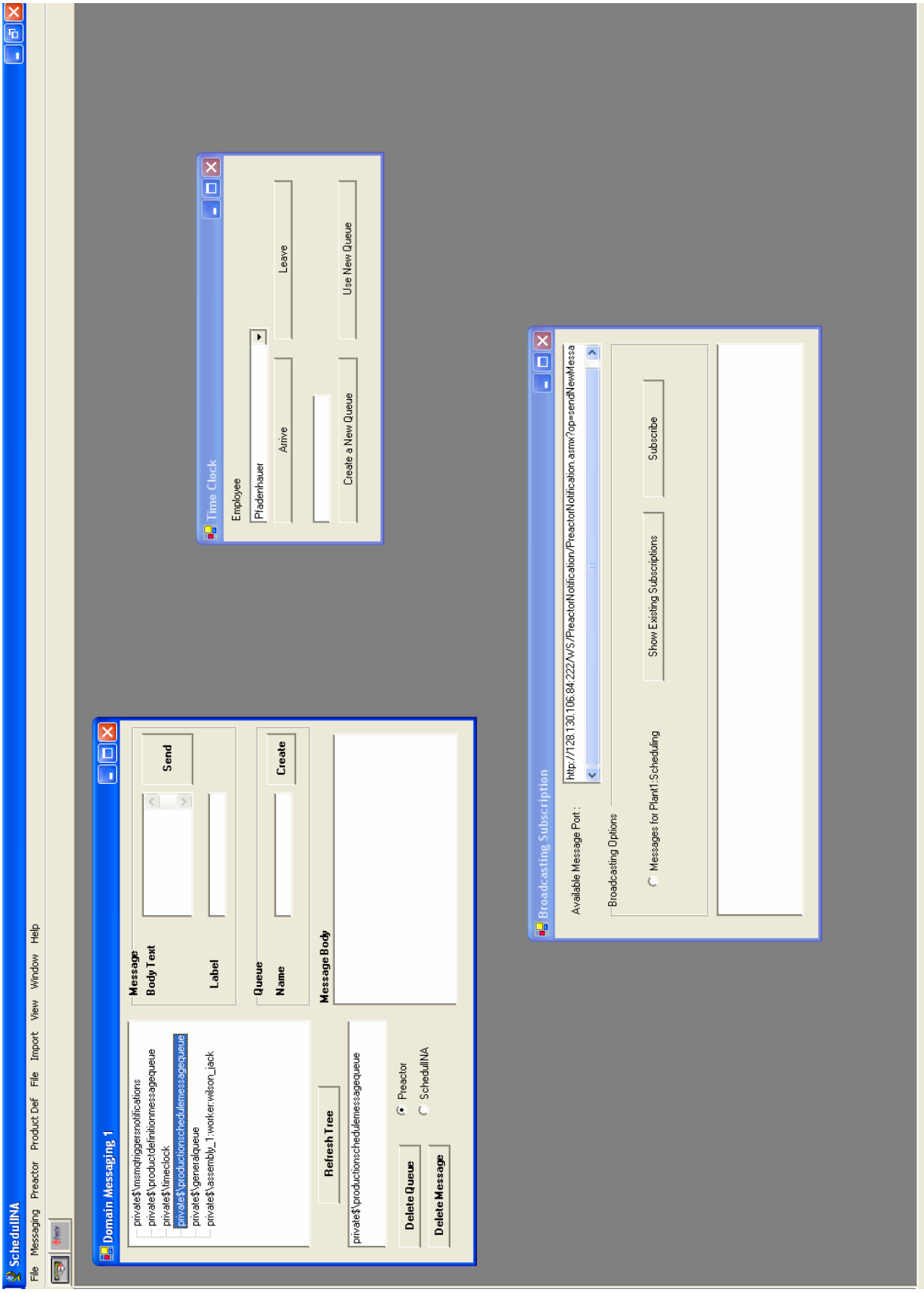


Figure 84. ScheduliNA application messaging related windows

The *Time Clock* window is for personnel availability monitoring. Hence in the system it is always visible if a worker is logged on and ready as a service provider. In the present configuration this information is only stored in a central queue, but a real time update of the UDDI would be feasible in the following step. A classification tag could then be used to indicate whether all the occurrences of a given endpoint were available in the system or not.

The specific endpoint available for this machine and thus for the user working on it can be seen in the *Broadcasting Subscription* window. Here, the user has the room to subscribe for certain messaging events directly in the UDDI. The example given here is the messaging traffic related to the Plant1:Scheduling partition. A subscription would mean that the user receives all messages broadcasted from the scheduling activities, e.g. the messages sent by the scheduleNewOrders BizTalk orchestration concerning files waiting for import.

The main window of the Preactor import and export functionality block shows Figure 85. On the left hand side the *Preactor Import* window contains the parameter setting options for file import to Preactor. This application communicates directly with the original Preactor database files by means of read / write operations. Compared to the standard import configuration capabilities in Preactor this GUI provides a faster mean to change the file names or the event scripts. Script definition itself has to be done in Preactor.

The same is true for the *Preactor Export* application. It has already been described that the import files are of comma separated file format, checked and prepared through preliminary process steps. The export file, the *Detailed Production Schedule*, is originally in the .csv format as well.

Thus the application provides the option to automatically create a XML duplicate, which can be sent by email too. Because this transformation is done within the Excel library, the resulting XML file still holds the Excel annotations. This is per se very useful, because the user can open both files with Excel and they will look identical.

But for further SOA processing, the Excel representation tags have to be erased. This file cleaning process step can be done by applying the *Export File Transformation* functionality. The output XML file is compliant to the DetailedProductionScheduling.xsd format and is ready for subsequent process steps. In the demo scenario, the *Production Execution Management* activity will route the released work operations to the correct resources.

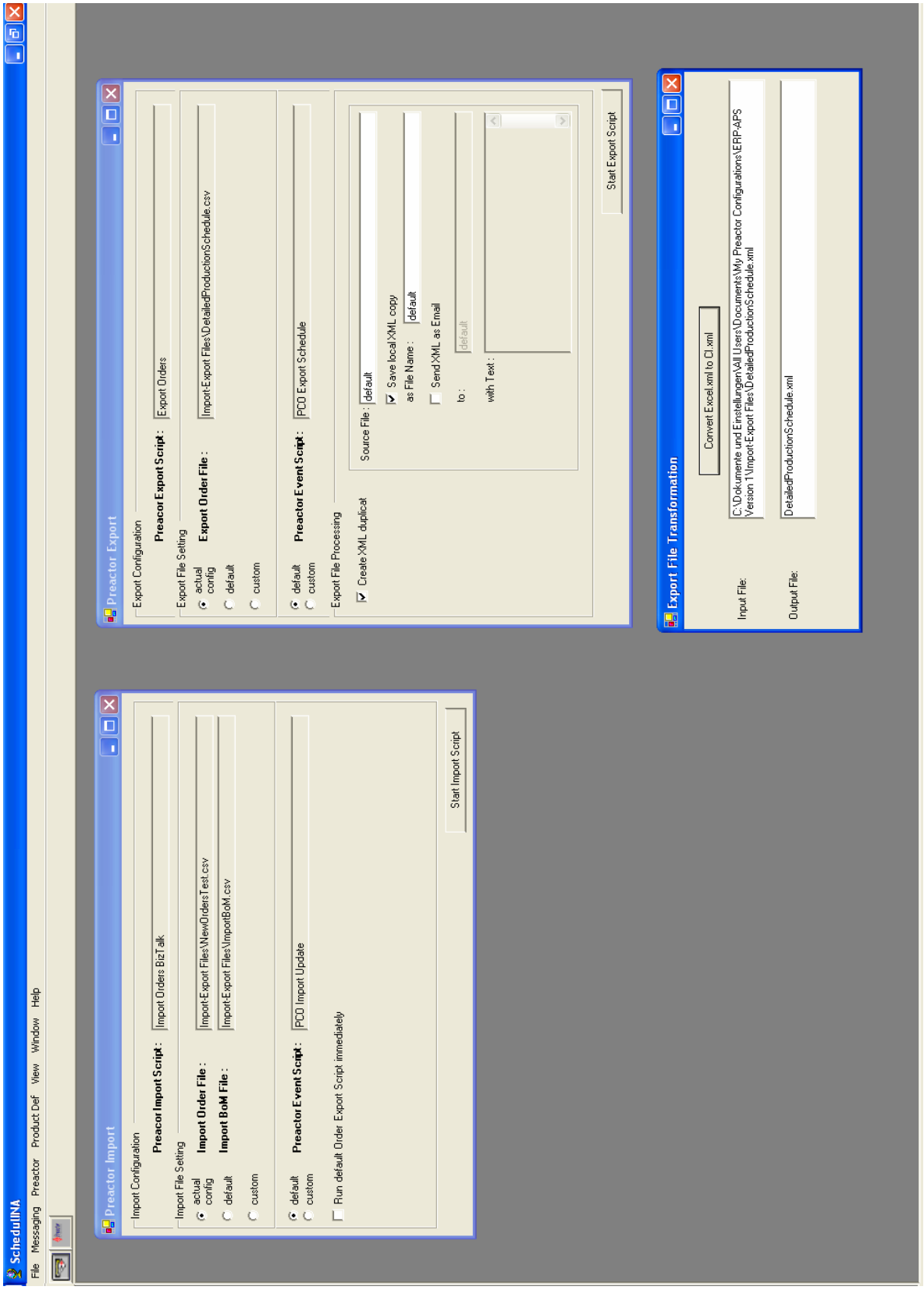


Figure 85. ScheduliNA import and export related windows

# 8. 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 modeling constructs.

The successful participation of IT and domain specialists in the evaluation phase as well as the reviewer comments for the publications (the latest was Pfadenhauer et al. 2006b) proved 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 modeling. Especially the issue of nested flows made some proprietary patterns necessary.

Next steps to come are some investigations regarding system dynamics. It would be interesting to know how the proposed 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. The concept of control loops was considered from the very beginning, the implementation probably within an enlarged demo scenario remains open for further research.

Another future focal point should 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.

To sum it up, it was proved that business and IT alignment for well defined domains like the shop floor in discrete manufacturing by means of system orientation, service orientation and modeling is successful as soon as a stringent framework like MDSA exists.

# 9. Appendix

## 9.1. Abbreviations

| | |
|---|---|
| ANSI | American National Standards Institute |
| API | Application Programming Interface |
| Appr. | Approximately |
| ARIS | Architektur integrierter Informationssysteme |
| ATP | Available to Promise |
| B2B | Business-to-Business |
| B2C | Business-to-Consumer |
| B2MML | Business to Manufacturing Markup Language |
| BoM | Bill of Material |
| BPEL | Business Process Execution Language |
| BPM | Business Process Management |
| BPML | Business Process Modeling Language |
| BPMN | Business Process Modeling Notation |
| BPR | Business Process Reengineering |
| CAD | Computer Aided Design |
| CAM | Computer Aided Manufacturing |
| CIM | Computer Integrated Manufacturing; Computation Independent Model |
| CIMOSA | Computer Integrated Manufacturing Open System Architecture |
| CORBA | Common Object Request Broker Architecture |
| COTS | Commercial off-the-shelf |
| DCOM | Distributed Component Object Model |
| DCS | Distributed Control System |

| | |
|---|---|
| DLL | Dynamic Link Library |
| DNC | Distributed Numeric Control |
| DoD | US Department of Defence |
| e.g. | example given |
| EA | Enterprise Architecture |
| EAI | Enterprise Application Integration |
| EDI | Electronic Data Interchange |
| ERP | Enterprise Resource Planning |
| ESB | Enterprise Service Bus |
| FAN | Field Area Network |
| FMS | Flexible Manufacturing System |
| FSO | File System Object |
| GERAM | Generalized Enterprise Reference Architecture and Methodology |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| IDE | Integrated Development Environment |
| IEC | International Electrotechnical Commission |
| IF | Interface |
| IIS | Internet Information Server |
| IMS | Information Management System |
| IS | Information System |
| ISA | Instrumentation, Systems and Automation Society |
| ISM | Information System Modeling |
| IT | Information Technology |
| KPI | Key Performance Indicator |
| MAS | Multi Agent System |

| | |
|---|---|
| MDA | Model Driven Architecture |
| MDSA | Model Driven Service Architecture |
| MES | Manufacturing Execution System |
| MIS | Management Information System |
| MOF | Meta Object Facility |
| MRP | Material Requirement Planning |
| MS | Microsoft |
| MSMQ | Microsoft Message Queuing |
| NC | Numeric Control |
| OLE | Object Linking and Embedding |
| OMG | Object Management Group |
| OO | Object Orientation |
| OOA | Object Oriented Analysis |
| OPC | OLE for Process Control |
| OS | Operating System |
| PC | Personal Computer |
| PERA | Purdue Enterprise Reference Architecture |
| PIM | Platform Independent Model |
| PLC | Programmable Logic Control |
| PPS | Production Planning System |
| PSM | Platform Specific Model |
| R&D | Research and Development |
| RFID | Radio Frequency Identification |
| SCADA | Supervisory Control And Data Acquisition |
| SCM | Supply Chain Management |
| SCOR | Supply-Chain Operation Reference model |

| | |
|---|---|
| SFTB | Shop Floor Tool Box |
| SOA | Service Oriented Architecture |
| SOAD | Service Oriented Analysis and Design |
| SOAP | Simple Object Access Protocol |
| UDDI | Universal Description, Discovery and Integration |
| UML | Unified Modeling Language |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| W3C | World Wide Web Consortium |
| WfMC | Workflow Management Coalition |
| WS | Web Service |
| WSDL | Web Services Description Languages |
| WS-SOA | Web Service – Service Oriented Architecture |
| XML | Extensible Markup Language |
| XPDL | XML Process Definition Language |
| XSD | XML Schema |
| XSLT | Extensible Stylesheet Language Transformation |

## 9.2. References

Aalst W.M.P van der, Barros A.P., ter Hofstede A.H.M. and Kiepuszewski B.: Advanced Workow Patterns. In O. Etzion and P. Scheuermann, editors, Fifth IFCIS International Conference on Cooperative Information Systems (CoopIS'2000), volume 1901 of Lecture Notes in Computer Science, pages 18-29, Eilat, Israel, September 2000. Springer-Verlag.

Aalst W.M.P. van der: Formalization and Verification of Event-driven Process Chains. Information and Software Technology, 41(10), 639-650, 1999

Aguilar-Savén R.S.: Business process modelling: Review and framework, *International Journal of Production Economics*, April, 2003

Alonso G., Casati F., Kuno H., Machiraju V.: Web Services – Concepts, Architectures and Applications, Springer, Berlin Heidelberg, 2004

Amsden J., Gardner T. et al.: Draft UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0, Version 1.1, IBM, June 9th 2003

Andrews W.: The Business of Web Services: Models and Opportunities; Gartner Teleconference, November 26th, 2003, GARTNER

ANSI / ISA-95.00.01-2000 Enterprise-Control System Integration Part 1: Models and Terminology, ISA Organization, 2000

ANSI / ISA-95.00.02-2001 Enterprise-Control System Integration Part 2: Object Model Attributes, ISA Organization, 2001

ANSI / ISA-95.00.03-2005 Enterprise-Control System Integration Part 3: Activity Models of Manufacturing Operations Management, ISA Organization, 2005

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

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

Barry D. K.: Web Services and Service-Oriented Architecture; The Savvy Manager's guide; San Francisco, Calif. : Morgan Kaufmann ; Oxford : Elsevier Science, 2003.

Bauer A.: Flexible control, *Manufacturing Engineer*, Volume 74, Issue 6, December, Pages 287 – 289, 1995

Beck K., Joseph J., Goldszmidt G.: Learn business process modeling basics for the analyst, IBM developerworks, Februar 22nd 2005, accessed at http://www-128.ibm.com/developerworks/webservices/library/ws-bpm4analyst/index.html on March 12th, 2005

Becker J., Kugeler M. and Rosemann M. (Editors): Prozessmanagement, Springer, Berlin Heidelberg New York, 2003

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

Blecker T.: Changes in Operations Management due to Internet based Production Concepts – An Institution Economical Perspective, Discussion paper of the College of Business Administration, University of Klagenfurt, No. 2003/02, Austria, June 2003

Booth A. W.: Object-Oriented Modeling for Flexible Manufacturing System, *International Journal of Flexible Manufacturing Systems*, Volume 10, Issue 3, 1998, Pages 301 – 314

Brittenham P., Cubera F., Ehnebuske D., Graham S.: Understanding WSDL in a UDDI registry, Part 1, IBM developerworks, September 2001, accessed at: http://www-128.ibm.com/developerworks/webservices/library/ws-wsdl/ on May 13th, 2004

Brown K., Reinitz R.: Web Services Architectures and Best Practices; IBM WebSphere Developer Technical Journal; IBM developerworks October 14th, 2003; accessed at http://www-128.ibm.com/developerworks/websphere/techjournal/0310_brown/brown.html on June 12th, 2004

Bruccoleri M., Noto La Diega S., Perrone G.: An Object-Oriented Approach for Flexible Manufacturing Control Systems Analysis and Design Using the Unified Modeling Language, *International Journal of Flexible Manufacturing Systems*, Volume 15, Issue 3, July 2003, Pages 195 – 216

Burbeck S.: The Tao of e-Business services; IBM developerworks October 1st, 2000; accessed at http://www-128.ibm.com/developerworks/webservices/library/ws-tao/ on March 23rd, 2002

Business Process Management Initiative (BPMI); BPMN Charter, Author: Notation Working Group Membership, BPMI Document Number NWG-2001-09-01R4, November 1st, 2001

Channabasavaiah, Kishore et al.: Migrating to a service-oriented architecture, Part 2; IBM developerworks December 16th, 2003; accessed at http://www-128.ibm.com/developerworks/webservices/library/ws-migratesoa2/ on June 12th, 2004

Colan M. (2004a): Service-Oriented Architecture expands the vision of Web services, Part I; IBM developerworks April 21st 2004; accessed at http://www-128.ibm.com/developerworks/library/ws-soaintro.html on February 27th, 2005

Colan M. (2004b): Service-Oriented Architecture expands the vision of Web services, Part II; IBM developerworks June 28th 2004; accessed at http://www-128.ibm.com/developerworks/library/ws-soaintro2.html on February 27th, 2005

Connelly D.: Open Applications Group Briefing, OAGIS Webcast on 18th April, 2005

Corsten H.: Produktionswirtschaft, 6. Auflage, Oldenbourg, München/Wien 1996

Dietzsch A.: Adapting the UML to Business Modeling's Needs – Experiences in Situational Method Engineering, In: J.M. Jézéquel, H. Hussmann and S. Cook, editors, UML 2002 – The Unified Modeling Language, Proc. Of the Int. Conference in Dresden, Germany, September/October, Springer Verlag Berlin Heidelberg, 2002, Pages 73-83

Di Salvo G., Johnsson C., Pazzini M.: A&D AS MES SIMATIC IT Production Suite V5, Release 1, February 28th, 2003, Siemens AG 2003

Dumas M., ter Hofstede A.H.M.: UML Activity Diagrams as a Workflow Specification Language, In: M. Gogolla, C. Kobryn, editors, UML 2001 – The Unified Modeling Language, Proc. Of the Int. Conference in Toronto, Canada, October, Springer Verlag, Berlin Heidelberg, 2001, pp76-90.

Dustdar S., Treiber M.: A View Based Analysis on Web service Registries. *Distributed and Parallel Databases*, Springer, 18, 147-171

Dustdar S.: Web Services Workflows - Composition, Coordination, and Transactions in Service-Oriented Computing, Concurrent Engineering: Research and Applications, Sage Publications, September 2004, p. 237-246

Endrei M. et al.: Patterns: Service-Oriented Architecture and Web Services; Redbooks, IBM International Technical Support Organization, April 2004

Erl T.: Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services; 2004 Pearson Education, Publishing as Prentice Hall PTR, New Jersey

ESPRIT consortium AMICE (Eds.): CIMOSA: Open System Architecture for CIM, ESPRIT Research Reports, Springer Verlag, Berlin Heidelberg, 1993

Favre-Bulle B., Zoitl A.: Zentralismus ist "out"!; a3 volt, 1-2, 2004; 25.Jahrgang

Förster A.: Pattern-basierte Modellierung von Geschäftsprozessen, Thesis at the Institute for Database- and Informationsystems, University of Paderborn, November 2002

Fredrik T.: Integrating Electrical Power Systems, From Individual to Organizational Capabilities, In: The Business of Systems Interation Ed.: Prencipe, Andrea; Davies, Andrew; and Hobday, Mike Oxford University Press, Uxford, New York, 2003, S56ff

Fröhlich P., Hu Z., and Schoelzke M.: Using UML for Information Modeling in Industrial Systems with Multiple Hierarchies, In: J.M. Jézéquel, H. Hussmann and S. Cook, editors, UML 2002 – The Unified Modeling Language, Proc. Of the Int. Conference in Dresden, Germany, September / October, Springer Verlag Berlin Heidelberg, 2002

Giaglis G.M.: A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques, International Journal of Flexible Manufacturing Systems, Volume 13, Issue 2, April 2001, 209 – 228

Grønmo R., Solheim I.: Towards Modeling Web Service Composition in UML, presented at The 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI-2004), Porto, Portugal, 2004

Haeckel S.H.: Leading on demand business-Executives as architects, IBM Systems Journal, Volume 42, Issue 3, 2003, pp405-413.

Hammer M., Champy J.: Business Reengineering, Die Radikalkur für das Unternehmen, 7.Aufl., Campus Fachbuch, 1996

Hitz M., Kappel, G.: UML@Work, dpunkt.verlag Heidelberg, 2nd edition 2003

IBM developerWorks Live; Streamline Business Processes with WebSphere Business Integration, Handout at the IBM Forum Vienna, June 24th, 2004

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

Interactive Objects Software GmbH: ArcStyler MDA-Business Transformer Modeling Style Guide For ARIS, For ArcStyler Version 3.x, November 2002

INTEROP D10.1: INTEROPERABILITY GLOSSARY, IST-508 011, D10.1 Final v1B, April 15th, 2005

INTEROP D12.1: The Methodology to implement services and develop take up actions towards SMEs, IST-508 011, D12.1 v5, November 25th, 2004

INTEROP D12.1: The Methodology to implement services and develop take up actions towards SMEs, IST-508 011, D12.1 Final v7, May 17th, 2005

INTEROP D4.1: Scientific Integration Conceptual Model and its application in INTEROP, IST-508 011, Version 5.4, November 19th, 2004

INTEROP D4.2: INTEROP 2nd Workplan, IST-508 011, D4.2 v1.4, December 2nd, 2004

INTEROP D6.1: Practices, principles and patterns for interoperability, Ed. David Chen, IST-508 011, Final Version 1.0, May 20th, 2005

INTEROP D8.1: State of the art and state of the practice including initial possible research orientations, IST-508 011, D8.1 v1.2, November 18th, 2004

INTEROP D9.1: State-of-the art for Interoperability architecture approaches, Model driven and dynamic, federated enterprise interoperability architectures and interoperability for non-functional aspects, IST-508 011, D9.1 v1.0, November 19th, 2004

Iung B.: From remote maintenance to MAS-based e-maintenance of industrial process, Journal of Intelligent Manufacturing, 14, pp 59-82, 2003

Januszewsji, K. (2002a): The Importance of Metadata: Reification, Categorization, and UDDI, Microsoft MSDN, September 2002, accessed at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnuddi/html/runtimeuddi1.asp on June 9th, 2004

Januszewski K. (2002b): Using UDDI at Run Time, Part II, Microsoft MSDN, May 2002, accessed at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnuddi/html/runtimeuddi1.asp on August 11th, 2005

Jin J.Q., Loftus M., Franks I.T.: A method for the acquisition of users requirements in discrete manufacturing cell systems, *Computer Integrated Manufacturing Systems*, Volume 11, Issue 3, 1998, Pages 229 – 242

Johnson S. B.: Systems Integration and the Social Solution of Technical Problems in Complex Systems, In: The Business of Systems Interation Ed.: Prencipe, Andrea; Davies, Andrew; and Hobday, Mike Oxford University Press, Uxford, New York, 2003, S35ff

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

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

Kalogeras A.P. et al.: Vertical Integration of Enterprise Industrial Systems Utilizing Web Services, in Sauter, T., Vasques, F.: Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems, Vienna, September 2004

Kedir Biadgligne A., Matyas K., Pfadenhauer K. (2004): Integrated MRP Heuristics for determining raw material purchasing lot in manufacturing enterprises, Annals of DAAAM for 2004 & Proceedings of the 15th International DAAAM Symposium, ISBN 3-901509-42-9, ISSN 1726-9679, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2004

Kittl B.: CAM: Computer Aided Manufacturing im betrieblichen Umfeld, WUV-Univ.-Verlag, Wien, 1993

Kloppmann et al.: WS-BPEL Extension for Sub-processes – BPEL-SPE, A Joint White Paper by IBM and SAP, September 2005, accessed at http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpelsubproc/ on March 9th, 2006

Kloppmann M., König D. et al.: Business process choreography in Websphere: combining the power of BPEL and J2EE, IBM Systems Journal, Volume 43, Issue 2, 2004, pp. 270 - 296

Kraemer D., Yendluri, P.: Realizing the Benefits of Implementing RosettaNet Implementation Framework (RNIF) Version 2.0, Technical White Paper 2002, accessed at http://www.rosettanet.org/Rosettanet/Doc/0/F9RGU5AMQBM4J66Q0BT84NQ247/RNIF2finalv3.pdf on September 14th, 2005

Leymann F., Roller D.: Using flows in information integration, IBM Systems Journal, Volume 41, Issue 4, 2002, pp 732 – 742

Leymann F., Roller D., Schmidt M.-T.: Web services and business process management, IBM Systems Journal, Volume 41, Issue 2, 2002, pp198 – 211

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

List B., Korherr B.: A UML 2 Profile for Business Process Modelling, Proceedings of the 1st International Workshop on Best Practices of UML (BP-UML 2005) at the 24th International Conference on Conceptual Modeling (ER 2005), Klagenfurt, Austria, 2005, Springer Verlag, Lecture Notes in Computer Science.

List B., Korherr B.: An Evaluation of Conceptual Business Process Modelling Languages, Proceedings of the 21st ACM Symposium on Applied Computing (SAC'06), April 2006, Dijon, France, ACM Press, 2006.

Mantell K.: From UML to BPEL, IBM developerworks, Sept 9th 2003, http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel/

Mick R.: The Alignment of Enterprise Architecture and Interoperability, ARC Insights, June 16th, 2005

Mielke C.: Geschäftsprozesse: UML-Modellierung und Anwendungsgenerierung, Spektrum Akademischer Verlag, Heidelberg Berlin, 2002

Miller J., Mukerji J. (Editors): MDA Guide Version 1.0.1, OMG, June 12th 2003

Mumm A.: Gestaltung der Prozesse zur Werkzeugversorgung in der spanenden Fertigung, Fachgespräch "Bohren und Fräsen im modernen Produktionsprozeß", Universität Dortmund, Tagung 21. / 22.5.97, Tagungsband, 1997, Pages 147 – 163

Ng A., Chen S. and Greenfield P.: An Evaluation of Contemporary Commercial SOAP Implementations, 5th Australasian Workshop on Software and System Architectures (AWSA2004), Pages 64-71, Melbourne, Australia (ISBN 0-85590-803-3), April 2004

Ng P.-W.: Effective Business Modeling with UML: Describing Business Use Cases and Realizations, the Rational edge, November 2002

OMG UML for SE draft 2003

OMG: Business Process Definition Metamodel, Request for Proposal, OMG document, 2003-01-06, http://www.omg.org/docs/bei/03-01-06.pdf

PABADIS, Plant Automation Based on Distributed Systems, Deliverable 6.3, IST-1999-60016, Work Package 6: Assessment and Evaluation, Task 6.3 Revolutionising Plant Automation, 2001

Pabadis.org: www.pabadis.org, accessed: 27.9.2004

Pfadenhauer K., Dustdar S., Kittl B. (2005c): Challenges and Solutions for Model Driven Web Service Composition, Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 3rd International Workshop on Distributed and Mobile collaboration (DMC), ISBN 0-7695-2362-5, pp 126-131, 13 - 15 June 2005, Linköping, Sweden, IEEE Computer Society Press.

Pfadenhauer K., Dustdar S., Kittl B. (2005d): Comparison of Two Distinctive Model Driven Web Service Orchestration Proposals, Proceedings of the 7th IEEE International Conference on E-Commerce Technology, http://cec05.in.tum.de/1st IEEE International Workshop on Service-oriented Solutions for Cooperative Organizations (SoS4CO'05), co-located with the, 19 July 2005, Munich, Germany.

Pfadenhauer K., Kittl B. (2003): Reactive scheduling for batch processors, Annals of DAAAM for 2003 & Proceedings of the 14th International DAAAM Symposium, ISBN 3-901509-34-8, ISSN 1726-9679, pp 357-358, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2003

Pfadenhauer K., Kittl B. (2004a): Modeling the shop floor for a SOA with UML, Proceedings of the 1st International Conference on Enterprise Systems and Accounting, Thessaloniki, Greece, 2004

Pfadenhauer K., Kittl B. (2004b): With a system approach towards a model driven service architecture for the shop floor, The 2004 Research Conference on Innovations in Information Technology, Dubai, UAE, 2004

Pfadenhauer K., Kittl B. (2004c): Flexibility in the shop floor by means of a model driven service architecture: a case study, Annals of DAAAM for 2004 & Proceedings of the 15th International DAAAM Symposium, ISBN 3-901509-42-9, ISSN 1726-9679, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2004

Pfadenhauer K., Kittl B. (2006a): Prozessmanagement in serviceorientierten Architekturen, ERP Management Issue 2, pp 40-42, 2006, GITO-Verlag

Pfadenhauer K., Kittl B., Dustdar S., Levy D. (2006b): Shop Floor Information Management and SOA, Accepted for publication at the 4th International Conference on Business Process Modelling, 2nd International Workhop on Enterprise and Networked Enterprises Interoperability (ENEI'06), Vienna, Austria, 4th September 2006

Pfadenhauer K., Kittl, B. (2005a). Why modeling tools and methodologies for a service oriented shop floor architecture are needed, CIM Issue 6, 7th International Conference on Computer Integrated Manufacturing, Intelligent Manufacturing Systems, ISBN 83-915011-3-2, pp 183-187, Gliwice, Poland

Pfadenhauer K., Kittl, B. (2005b). Why model driven service orientation is the future of shop floor integration, Proceedings of the COMMENT 2005 Worldwide Congress on Materials and Manufacturing Engineering and Technology, ISBN 83-89728-13-3, Editor L.A. Dobrzanski, Gliwice, Poland

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

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

Prencipe A.: Corporate Strategy and Systems Integration Capabilities, Managing Networks in Complex Systems Industries, In: The Business of Systems Interation Ed.: Prencipe, Andrea; Davies, Andrew; and Hobday, Mike Oxford University Press, Uxford, New York, 2003, S114ff

Rational: Business Modeling with the UML and Rational Suite AnalystStudio, A Rational Software White Paper, 2001, accessed at www.rational.com on February 12th, 2004

Ritter G.: Cell Integrator, master thesis at the Institute of Production Engineering, Vienna University of Technology, May 2003

Rosenberg F., Dustdar S.: Business Rules Integration in BPEL – A Service-Oriented Approach. 7th International IEEE Conference on E-Commerce Technology (CEC 2005), 19 - 22 July 2005, Munich, Germany

RosettaNet: PIP Specification, Cluster 7: Maufacturing, Segment B: Manage Manufacturing WO & WIP, 7B1: Distribute Work in Process, Release 01.00.00A, September 14th, 2005

RosettaNet: RosettaNet and WebServices, An Executive-Level View of RosettaNet and an Emerging Model for Application-Based B2B Commerce, Technical White Paper 2003, accessed at http://www.rosettanet.org/Rosettanet/Doc/0/IP0QL046K55KFBSJ60M9TQCPB3/RosettaNet%20Web%20ServicesFINAL%20.pdf on September 14th, 2005

Schmidt M.-T. et al.: The Enterprise Service Bus: Making service-oriented architecture real; IBM Systems Journal, Volume 44, Number 4, 2005

Selic B.: Unified Modeling Language version 2.0, In support of model-driven development, IBM developerworks, March 2005, accessed at http://www-128.ibm.com/developerworks/rational/library/05/321_uml/ on August 23rd, 2005

Shi D., Daniels R.L.: A survey of manufacturing flexibility: Implications for e-business flexibility, IBM Systems Journal, Volume 42, Issue 3, 2003, pp 414 – 427

Siemens: Simatic – Integration von MES-Funktionen in PCS7 mit Simatic IT, Handbuch, Siemens AG Automation & Drives, Ausgabe 10/2005, A5E00732179-01

Siemens: Simatic RF600, RFID system in the UHF range for logistics and distribution, Brochure – Preliminary Information, 2006, accessed at http://www.automation.siemens.com/rfid/html_76/download_broschueren.htm on March 17th, 2006

Sihn W., Graupner T.-D.: e-Industrial Services for Manufacturing Systems: Differentiation Through Internet Services, Proceedings of the 36th CIRP International Seminar on Manufacturing Systems, Saarbrücken, Germany, 2003

Sinogas P., Vasconcelos A., Caetano A., Neves J., Mendes R., Tribolet J.: Business Processes Extensions to UML Profile for Business Modeling. Proceedings of the International Conference on Enterprise Information Systems, 2001.

Skogan D., Grønmo R., Solheim I.: Web Service Composition in UML, The 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC), Monterey, California, 2004

Sprott D. (2004a); Service Oriented Architecture: An Introduction for Managers; CBDI Report, CBDI Forum Limited 2004

Sprott D. (2004b); Wilkes, L.: Understanding Service-Oriented Architecture; CBDI Forum, January 2004, accessed at http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj1soa.asp on April 24th, 2005

Steinmueller W. E.: The Role of Technical Standards in Coordinating the Division of Labour in Complex System Industries, In: The Business of Systems Interation Ed.: Prencipe, Andrea; Davies, Andrew; and Hobday, Mike Oxford University Press, Uxford, New York, 2003, S133ff

Supply Chain Council: Supply-Chain Operations Reference-model, SCOR Version 6.1 Overview, accessed at: www.supply-chain.org on March 2nd, 2005

Tian G.Y., Yin G., Taylor D.: Internet-based manufacturing: A review and a new infrastructure for distributed intelligent manufacturing, Journal of Intelligent Manufacturing, 13, pp 323-338, 2002

Voigt H.: Modell-basierte Analyse von ausführbaren Geschäftsprozessen für Web Services, Thesis at the Institute for Database- and Informationsystems, University of Paderborn, September 2003

Wagner T., Blumenau J.-C.: The Digital Factory, more than a Planning Environment, Proceedings of the 36th CIRP International Seminar on Manufacturing Systems, Saarbrücken, Germany, 2003

Walford R.: Business Process Impementation for IT Professionals and Managers, Artech House, Boston, London, 1999

Westkämper E., Jendoubi L.: Smart Factories – Manufacturing Environments and Systems of the Future, Proceedings of the 36th CIRP International Seminar on Manufacturing Systems, Saarbrücken, Germany, 2003

White S. A.: Process Modeling Notations and Workflow Patterns; IBM, January 2004, accessed at http://www.bpmn.org/Documents/Notations%20and%20Workflow%20Patterns.pdf on November 22nd, 2005

Whitman L., Huff B.: On the Use of Enterprise Models, International Journal of Flexible Manufacturing Systems, Volume 13, Issue 2, April, pp195 – 208, 2001.

Wiendahl H.-H., Mussbach-Winter U., Kipp R.: Marktspiegel Business Software MES – Fertigungssteuerung 2004/2005, Ed.: Frauenhofer Institute for Production Engineering and Automation IPA, Stuttgart, and Trovarit AG, Aachen, 2004

Wollschlaeger M., Bangemann T.: Maintenance Portals in Automation Networks – RequirementsStructures and Model for Web-based Solutions, In: Sauter T., Vasques F.: Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems, September 2004

Woods D. and Word J.: SAP NetWeaver for dummies, Wiley Publishing, Indianapolis, Indiana, USA, 2004

Workflow Management Coalition (WfMC); Documents and Interfaces 2005, accessed at http://www.wfmc.org/standards/docs.htm on February 15th, 2006

World Wide Web Consortium (W3C): Web Services Architecture – W3C Working Group Note, February 11th, 2004; accessed at http://www.w3.org/TR/ws-arch/ on February 14th, 2006

World Wide Web Consortium (W3C): Web Services Description Languages (WSDL) 1.1, W3C Note 15 March 2001; accessed at http://www.w3.org/TR/wsdl on January 14th, 2003

World Wide Web Consortium (W3C): Web Services Glossary – W3C Working Draft, August 8th, 2003; accessed at http://www.w3.org/TR/2003/WD-ws-gloss-20030808/ on September 14th, 2003

Zhang J., Gu J., Li P. and Duan Z.: Object-oriented modeling of control system for agile manufacturing cells, *International Journal of Production Economics,* Volume 62, Issues 1-2, May, Pages 145 – 153

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

## 9.3. List of Figures

## 9.4. List of Tables

## 9.5. Curriculum Vitae

Name: Dipl.-Ing. Konrad Pfadenhauer

Date of birth: April 24th, 1977

Nationality: Austria

Adress (office): Karlsplatz 13, 1040 Vienna, Austria

Email: Pfadenhauer@mail.ift.tuwien.ac.at

| | |
|---|---|
| 10.1995 – 06.2002 | Vienna University of Technology, Austria, Study „Industrial Engineering" Thesis: "Logistics optimization in batch processes: a case study" |
| 10.1999 – 06.2000 | University of St. Gallen, Switzerland, Visiting Student (investment, finance, operations research) |
| 09.2002 – 08.2006 | Research Assistant, Vienna University of Technology, Austria, Institute for Production Engineering |
| | Ph.D. project topic: Modeling the shop floor for a service oriented architecture |
| 07.2005 – 12.2005 | Visiting Academic at the School of Electrical and Information Engineering, University of Sydney (Australia) |
| 09.2004 – 06.2006 | Scientific head of education for the "Pre-Production Management" university diploma program |
| Courses Taught | Production Engineering (basics and advanced) |
| Curriculum Developed | Designed and developed university diploma programs titled "Pre-Production Management", "Production Engineering", "Industrial Engineering" and "Design Engineering". These courses are offered as two year in-service programs to industry professionals. |